MSC8156E Reference Manual

Six Core Digital Signal Processor with Security

MSC8156ERM
Rev 2, June 2011

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

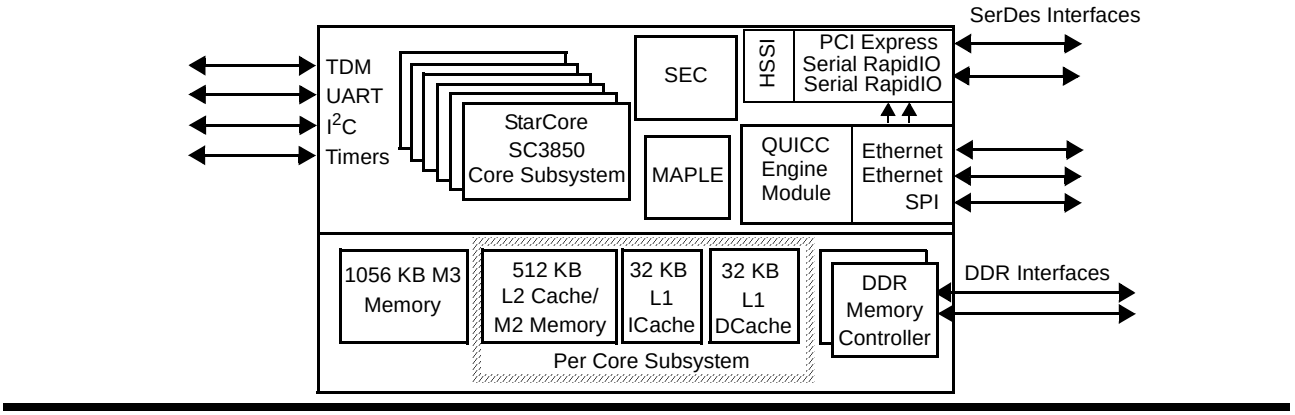
MSC8156ERM
Rev. 2
6/2011

Contents

17 PCI Express Controller

About This Book

The MSC8156E device is the fourth generation of Freescale high-end multicore DSP devices. It builds upon the proven success of the previous multicore DSPs and is designed to bolster the rapidly changing and expanding wireless markets, such as 3GPP, TD-SCDMA, 3G-LTE, and WiMAX. Its tool suite provides a full-featured development environment for C/C++ and assembly languages as well as ease of integration with third-party software, such as off-the-shelf libraries and a real-time operating system. The MSC8156E device includes six DSP core subsystems, a large internal memory subsystem and DDR memory controller for external memory, and a variety of communication processors and interfaces.



Six DSP Core Subsystems

Each DSP core subsystem includes an SC3850 DSP core, a 32 KB 8-way level 1 ICache, a 32 KB 8-way level 1 DCache, 512 KB level 2 cache configurable as M2 memory, a memory management unit, an embedded programmable interrupt controller (EPIC) with up to 256 interrupts and 32 priority levels, two general-purpose 32-bit timers, an on-chip emulator (OCE), a debug and profiling unit (DPU), a JTAG test access port (TAP), and two low-power operating modes (Wait and Stop). Interface from the cores to the memories and external interfaces is through a chip-level arbitration and switching system (CLASS).

Memory Subsystem

The memory subsystem includes 1056 KB of shared M3 memory, two DDR-SDRAM controller to access up to 0.5 GB of DDR2/3 external memory, and a 32-channel direct memory access (DMA) controller optimized for DDR-SDRAM.

MAPLE-B

The multi-accelerator platform engine (MAPLE-B) provides Turbo decoding, Viterbi decoding, FFT and DFT acceleration.

Security Engine (SEC)

The optional SEC has an internal bus controller, 4 data channels, 6 execution units, and a shared random number generator for communications security applications encryption/ decryption support.

Communications Processors and Interfaces

Includes a PCI Express interface, two serial RapidIO® interfaces, four 512-channel (256 transmit and 256 receive) TDM interfaces, a UART interface, an I^2C interface, eight timer input/outputs, and a QUICC Engine module with two 1000Base-T Ethernet controllers and an SPI. In addition, the global interrupt controller (GIC) consolidates all chip-maskable and non-maskable interrupts and routes them to $\overline{\text{NMI\_OUT}}$, $\overline{\text{INT\_OUT}}$, and to the cores. The hardware semaphores allow initiators to protect and reserve the system hardware resources.

Before Using This Manual—Important Note

This manual describes the structure and function of the MSC8156E device. The information in this manual is subject to change without notice, as described in the disclaimers on the title page of this manual. As with any technical documentation, it is your responsibility as the reader to ensure that you are using the most recent version of the documentation. For more information, contact your sales representative.

Before using this manual, determine whether it is the latest revision and whether there are errata or addenda. To locate any published errata or updates associated with this manual or this product, refer to the Freescale web site. The address for the web site is listed on the back cover of this manual.

Audience and Helpful Hints

This manual is intended for software and hardware developers and applications programmers who want to develop products with the MSC8156E. It is assumed that you have a working knowledge of DSP technology and that you may be familiar with Freescale products based on StarCore technology.

For your convenience, the chapters of this manual are organized to make the information flow as predictably as possible. When feasible, the information in each chapter follows this general sequence:

- General description, block diagram, features, and architecture
- Functional description with operating modes and example applications and programming
- Programming Model (registers)

In chapters that include a Programming Model section, this section is the last one in the chapter, or module subsection for those chapters that include multiple modules, and describes all registers for the module discussed. The Programming Model section begins with a bulleted overview of the registers that includes the page number where the description of each register begins.

Notational Conventions and Definitions

This manual uses the following notational conventions:

mnemonics	Instruction mnemonics appear in lowercase bold.
COMMAND names	Command names are set in small caps, as follows: GRACEFUL STOP TRANSMIT or ENTER HUNT MODE.
italics	Book titles in text are set in italics, as are cross-referenced section titles. Also, italics are used for emphasis and to highlight the main items in bulleted lists.
0x	Prefix to denote a hexadecimal number.
0b	Prefix to denote a binary number.
REG[FIELD]	Abbreviations or acronyms for registers or buffer descriptors appear in uppercase text. Specific bits, fields, or numeric ranges appear in brackets. For example, ICR[INIT] refers to the Force Initialization bit in the host Interface Control Register.
ACTIVE HIGH SIGNALS	Names of active high signals appear in sans serif capital letters, as follows: TT[04], TSIZ[0–3], and DP[0–7].
$\overline{\text{ACTIVE LOW SIGNALS}}$	Signal names of active low signals appear in sans serif capital letters with an overbar, as follows: $\overline{\text{DBG}}$, $\overline{\text{AACK}}$, and $\overline{\text{EXT\_BG[2]}}$.
x	A lowercase italicized x in a register or signal name indicates that there are multiple registers or signals with this name. For example, BRCG*x* refers to BRCG[1–8], and M*x*MR refers to the MAMR/MBMR/MCMR registers.

On the MSC8156E device, the SC3850 cores are 16-bit DSP processors. The following table shows the SC3850 assembly language data types. For details, see the *StarCore SC3850 DSP Core Reference Manual*.

Name	SC3850
Byte/Octet	8 bits
Half Word	8 bits
Word	16 bits
Long/Long Word/2 Words	32 bits
Quad Word/4 Words	64 bits

The following table lists the SC3850 C language data types recognized by the StarCore C compiler. For details, see the *StarCore SC100 C Compiler User's Manual (MNSC100CC/D)*.

Name	Size
char/unsigned char	8 bits
short/unsigned short	16 bits
int/unsigned int	16 bits

Name	Size
fractional short	16 bits
long/unsigned long	32 bits
fractional long	32 bits
pointer	32 bits

Conventions for Registers

The Programming Model section of each chapter includes a register bit table for each register in that module, as well as a table describing each bit in the register. The register bit table not only shows the names and positions of the bits/bit fields but also their reset value, value after boot, and their type (Read/Write). For registers that are not changed by the system boot, no boot line is listed. The register address is shown with the register name and mnemonic. Reserved bits/fields are indicated with a long dash (—). In the RSR shown below, all of the bits are read/write (R/W). Other registers may include read-only (R) and write-only (W) bits. Notice that the least significant bit (LSB) is 0, or big-endian order.

RSR Reset Status Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RSTSRC			—					RIO	SW1	SW2	SW3	—			BSF
Type	R/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
	—		SWSR	SWHR	—	JPO	JH	JS	—				SW	–	SRS	HRS
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Organization

Following is a summary and a brief description of the chapters of this manual:

- **Chapter 1, *Overview***. Features, descriptive overview of main modules, configurations, and application examples.
- **Chapter 2, *SC3850 Core Overview***. Target markets, features, overview of development tools, descriptive overview of main modules.
- **Chapter 3, *External Signals***. Identifies the external signals, lists signal groupings, including the number of signal connections in each group, and describes each signal within a functional group.
- **Chapter 4, *Chip-Level Arbitration and Switching System (CLASS)***. Describes the system switch fabric that allows multi-initiator access to the internal memory and devices and enables high-bandwidth internal data transfers with few bottlenecks.

- **Chapter 5, *Reset***. Covers reset sources, causes, and configurations; gives examples of different reset configuration scenarios, including systems with multiple MSC8156E devices.
- **Chapter 6, *Boot Program***. Describes the bootloader program that loads and executes source code to initialize the MSC8156E after it completes a reset sequence and programs its registers for the required mode of operation. This chapter covers selection of bootloader modes, normal sequence of events for bootloading a source program, and booting in a multi-processor environment.
- **Chapter 7, *Clocks***. Contains an overview of the MSC8156E clock modules.
- **Chapter 8, *General Configuration Registers***. Contains a detailed description of the general configuration registers.
- **Chapter 9, *Memory Map***. Defines the address spaces for all MSC8156E modules; includes cross references to all registers discussed.
- **Chapter 10, *MSC8156E SC3850 DSP Subsystem***. Describes the structure of the DSP core subsystem, which includes the SC3850 core, the instruction cache (ICache), the data cache (DCache), L2/M2 memory, memory management unit (MMU), two 32-bit timers, the embedded programmable interrupt controller (EPIC), and the on-chip emulator (OCE).
- **Chapter 11, *Internal Memory Subsystem***. Describes the structure and operation of the L1 ICache, L1 DCache, L2/M2 memory, and M3 memory.
- **Chapter 12, *DDR SDRAM Memory Controller***. Describes the how the memory controller interface works and how to program it. This interface increases the efficiency of accesses through the DDR memory controller to external DDR memory.
- **Chapter 13, *Interrupt Handling***. Discusses the interrupt controllers that provide maximum flexibility in handling MSC8156E interrupts, enabling interrupts to be handled by the SC3850 cores internally, by an external host, or by a combination of the two; also discusses source priority schemes.
- **Chapter 14, *Direct Memory Access (DMA) Controller***. Describes the different DMA operating modes, transfer types, and buffer types. The chapter also gives procedures for programming different types of transfers. The multi-channel DMA controller includes hardware support for up to 16 time-multiplexed channels including buffer alignment. The DMA controller supports flyby transactions on either bus. and enables hot swaps between channels, by using time-multiplexed channels that impose no cost in clock cycles.
- **Chapter 15, *High Speed Serial Interface (HSSI) Subsystem***. Describes subsystem that supports and multiplexes the Serial RapidIO, PCI Express, and SGMII signals across the dual SerDes PHY ports and how the dedicated DMA controllers support the serial RapidIO interfaces and PCI Express interface and how to program them.
- **Chapter 16, *Serial RapidIO Controller***. Describes the how the serial RapidIO interfaces work and how to program them.

- **Chapter 17, *PCI Express Controller***. Describes the how the PCI Express interface works and how to program it.
- **Chapter 18, *QUICC Engine Subsystem***. Describes the QUICC Engine module, the Ethernet controllers, and the serial peripheral interface (SPI). Detailed information is referenced in the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM).
- **Chapter 19, *TDM Interface***. Describes the four TDM interfaces. Each can handle up to 256 bidirectional channels. The interfaces support the serial bus rate and format for most standard TDM buses, including T1 and E1 highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates.
- **Chapter 20, *UART***. Describes the UART interface, which is a full-duplex serial port used to communicate with other devices.
- **Chapter 21, *Timers***. Discusses the 32 identical 16-bit general-purpose timers residing in two timer modules (A and B) that each have their set of configuration registers.
- **Chapter 22, *GPIO***. Discusses the thirty-two GPIO signals. Sixteen of the signals can be configured as external interrupt inputs. Each pin is multiplexed with other signals and can be configured as a general-purpose input, general-purpose output, or a dedicated peripheral pin.
- **Chapter 23, *Hardware Semaphores***. Describes the function and programming of the hardware semaphores, which control resource sharing.
- **Chapter 24, *I2C***. Describes the I^2C interface. which allows the MSC8156E to boot from a serial EEPROM device.
- **Chapter 25, *Debugging, Profiling, and Performance Monitoring***. Includes aspects of the JTAG implementation that are specific to the SC3850 core and should be used with the supporting **IEEE**® Std. 1149.1™ documentation. The discussion covers the items that the standard requires to be defined and provides additional information specific to the MSC8156E implementation. Also includes debugging resources available in the SC3850 DSP core subsystem, including the OCE modules, and L2 ICache module.
- **Chapter 26, *Multi Accelerator Platform Engine, Baseband (MAPLE-B)***. Describes the architecture, function, and register and memory structures used by the multi-accelerator platform engine (MAPLE-B) for Turbo decoding, Viterbi decoding, Fast Fourier Transform and Discrete Fourier Transform acceleration.
- **Chapter 27, *Security Engine (SEC)***. Describes the architecture, function, and register and memory structures used for security algorithm processing.

Other MSC8156E Documentation

You can find the following documents on the Freescale Semiconductor web site listed on the back cover of this manual.

- *MSC8156E Data Sheet (*MSC8156E*)*. Details the signals, AC/DC characteristics, clock signal characteristics, package and pinout, and electrical design considerations of the MSC8156E device.
- *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM). Describes all functional blocks supported by the QUICC Engine technology, provides detailed programming registers and guidelines, and indicates which QUICC Engine blocks and functionality are supported by specified Freescale products.
- *Application Notes*. Cover various programming topics related to the StarCore DSP core and the MSC8156E device.

Further Reading

The following documents are available with a signed non-disclosure agreement (see your Freescale representative or distributor for details):

- *SC3850 DSP Core Reference Manual*. Covers the SC3850 core architecture, control registers, clock registers, program control, on-chip emulator (OCE), and instruction set.
- *SC3850 DSP Core Subsystem Reference Manual*. Covers the SC3850 DSP core subsystem which includes an SC3850 DSP core, a memory management unit (MMU), and instruction channel with L1 ICache, a data channel with L1 DCache, an embedded programmable interrupt controller (EPIC), real-time debug support with the core OCE and a JTAG interface and a debug and profiling unit (DPU), and a dual timer.

Document Change History

Revision	Date	Change Description
0	Sep 2010	Initial public release
1	Nov 2010	• **Chapter 4, *Chip-Level Arbitration and Switching System (CLASS)***. – Removed all references to CLASS MBus Target Configuration Registers (C0MTCRn), CLASS Start Address Decoder 7 (C0SAD7), CLASS End Address Decoder 7 (C0EAD7), and CLASS Attributes Decoder 7 (C0ATD7). – Added **4.3**, *MSC8151 Initiator CLASS Access Priorities*. • **Chapter 6, *Boot Program*** – Updated **Figure 6-1**. – Updated Steps 10 and 11 in **Section 6.1.4**, *Multi Device Support for the I2C Bus*, on page 6-4. – Updated **Section 6.2.4.1**, *Serial RapidIO Without I2C Support*, on page 6-21. – Updated **Figure 6-10**. • **Chapter 7, *Clocks*** – Added Clock Mode 19 to **Table 7-1** and **Table 7-2**. – Fixed RPTE field in figure layout in **Section 7.2.2**, *Clock General Purpose Register 0 (CLK_GPR0)*, on page 7-5. • **Chapter 9, *Memory Map*** – Deleted C0MTCRn, C0SAD7, C0EAD7, C0ATD7, P0PCR, and P1PCR from **Table 9-7**. • **Chapter 13, *Interrupt Handling*** – Changed core mesh interrupts from level-only to level or edge in **Table 13-6**. • **Chapter 16, *Serial RapidIO Controller*** – Converted list of registers to **Table 16-44**. – Removed references to Port [0–1] Physical Configuration Registers (PnPCR). • **Chapter 17, *PCI Express Controller*** – Added new section **Section 17.4.1.9.15**, *PCI Express ACK Replay Timeout Register (PEX_ACK_REPLAY_TIMEOUT)—0x434*, on page 17-116. • **Chapter 25, *Debugging, Profiling, and Performance Monitoring*** – Removed references to trigger mode including PMLCB0. – Updated the settings values for DP_TC[CSS] in **Table 25-32**. – Updated the list of performance monitor events in **Table 25-38**.
2	Jun 2011	• **Chapter 4, *Chip-Level Arbitration and Switching System (CLASS)*** – Remove the reference to Target Switch events from **Table 4-2** • **Chapter 12, *DDR SDRAM Memory Controller*** – Updated **Figure 12-4** to change length of arrows between banks and total size of each bank (from 64 Mbytes to 256 Mbytes). – Changed heading of fourth and fifth columns in **Table 12-3** from 32-bit to 64-bit. • **Chapter 15, *High Speed Serial Interface (HSSI) Subsystem*** – Added OCN-toMBus control registers in **Section 15.8**, *HSSI Programming Model*. • **Chapter 16, *Serial RapidIO Controller*** – Corrected DI value in DIDCAR in field description. • **Chapter 25, *Debugging, Profiling, and Performance Monitoring*** – Corrected JTAG ID value. • **Chapter 26, *Multi Accelerator Platform Engine, Baseband (MAPLE-B)*** – Corrected the values and meanings of the DOSBY and DOSBI bits in **Section 26.3.2.1.4**, *Output Data Structure*.

Overview 1

The MSC8156E device is the fourth generation of Freescale high-end multicore DSP devices that target the communications infrastructure and delivers the industry's highest level of performance and integration. It builds upon the proven success of the previous multicore DSPs and is designed to bolster the rapidly changing and expanding wireless markets, such as 3GPP, TD-SCDMA, 3G-LTE, and WiMAX. The MSC8156E is carefully optimized for minimal cost, power, and area per channel. The highly flexible, fully-programmable and powerful MSC8156E broadband wireless access DSP offers tremendous processing power while maintaining a competitive price and high performance.

The highly integrated MSC8156E DSP device includes the following:

- Six StarCore® SC3850 DSP subsystems each running at up to 1 GHz with an architecture optimized for wireless applications.
- Two DDR2/3 memory controllers high-speed industry-standard memory interface.
- Multi-Accelerator Platform Engine for Baseband (MAPLE-B) supports hardware acceleration for Turbo decoding, Viterbi decoding, Fast Fourier Transform and Discrete Fourier Transform algorithm processing.
- High-Speed Serial Interface (HSSI) subsystem that supports
  - Two serial RapidIO interfaces
  - Two Gigabit serial Ethernet interfaces
  - One PCI-Express controller
- QUICC Engine RISC-based subsystem to guarantee reliable data transport over packet networks while significantly off loading such processing from the DSP cores that supports:
  - Two gigabit Ethernet controllers with RGMII and SGMII support
  - One SPI
- Four 256-channel time-division multiplexing (TDM) interfaces
- 16 bidirectional channels DMA controller
- UART interface
- I^2C interface
- Security Engine Core (SEC) to support multiple security algorithms and networking protocols, including IPSec and accelerates data plane encryption/decryption and code protection with minimal DSP cores intervention.

1.1 Features

The MSC8156E includes the following features:

- StarCore DSP subsystem. The DSP subsystem includes:
  - StarCore SC3850 core
    - Running at up to 1 GHz
    - Up to 8000 16-bit MMACS. A MAC operation includes a multiply-accumulate command with the associated data moves and a pointer update.
    - Backwards binary compatible with the SC140 and SC3400 architectures.
    - Data Arithmetic and Logic Unit (DALU) containing 4 ALUs, each capable of performing 2 16 × 16 multiply accumulate operations, effectively doubling the performance of convolution-based kernels relative to the SC3400 core
    - New instructions double the performance of complex and extended precision multiplication.
    - Address Generating Unit (AGU) containing 2 Address Arithmetic Units (AAU)
    - Up to six instructions executed in a single clock cycle: 4 DALU and 2 AGU instructions
    - Variable-length Execution Set (VLES) execution model.
    - 16 data registers, 40 bits each; 27 address registers, 32 bits each.
    - Hardware support for fractional and integer data types.
    - Four hardware loops with near-zero cycle overhead
    - Very rich 16-bit wide orthogonal instruction set.
    - Application specific instructions for Viterbi and Multimedia.
    - Special SIMD (Single instruction, multiple data) instructions working on 2-word or 4-byte operands packed in a register, enabling to perform 2 to 4 operations per instruction (8 to 16 operations per VLES)
    - New dedicated instructions accelerate FFTs enabling a 40% cycle count reduction and improved SNR
    - User and Supervisor privilege levels, supporting a protected SW model
    - New instructions and features to improve control code performance
    - Precise exceptions for memory accesses enabling good RTOS support and Soft Error corrections
    - Branch Target Buffer (BTB) for acceleration of change of flow operations
  - L1 ICache:
    - 32 Kbytes
    - 8 ways with 16 lines of 256 bytes per line
    - Multi-task support

- Real-time support through locking flexible boundaries
- Line pre-fetch capability
- Software coherency support
- Software pre-fetch support by core instructions

— L1 DCache:
- 32 Kbytes
- 8 ways with 16 lines of 256 bytes per line
- Capable of serving two data accesses in parallel (XA, XB)
- Multi-task support
- Real-time support through locking flexible boundaries
- Software coherency support
- Writing policy programmable per memory segment as either write-back or write-through
- 0.25 Kbytes Write-back Buffer (WBB)
- Six 64-bit entry WTB
- Line pre-fetch capability
- Software pre-fetch, synchronize, and flush support by core instructions

— Unified L2 Cache/M2 Memory:
- 512 Kbyte
- 8 ways with 1024 indexes and a 64 byte line
- Physically addressed
- Dynamically configured as a DMA accessible M2 Memory
- Maximum user flexibility for real time support through address partitioning of the cache
- Support various write policies and methods to reduce cache inclusiveness
- Multi-channel, two dimensional software pre-fetch support
- Software coherency support with seamless transition from L1 cache coherency operation.

— Memory management unit (MMU):
- Highly flexible memory mapping capability
- Provides virtual to physical address translation
- Provides task protection
- Supports multi-tasking
- Supports precise interrupts. Enabling to have an open RTOS.

— Debug and Profiling Unit (DPU) block:
  - Supports the debugging and profiling of the platform in cooperation with the OCE Block
  - Supports various breakpoint and event counting options
  - Supports real-time tracing to the main memory with the Trace Write Buffer (TWB)
— Extended programmable interrupt controller (EPIC)
  - 256 interrupts
  - 32 priority levels with NMI support
— Two general-purpose 32-bit timers
— Low-power design with the following modes of operation:
  - Wait processing state for peripheral operation
  - Stop processing state
— ECC/EDC support.

■ Multi Accelerator Platform Engine for Baseband (MAPLE-B)

— Programmable System Interface
  - Software friendly buffer descriptor based handshake and task assignment.
  - Support for high priority and low priority tasks via multiple descriptor rings.
  - Processing elements management and scheduling.
  - Two master buses for data transfers from/to the system memory at total throughput up to 50 Gbps.
  - One slave bus for accessing MAPLE-B internal memories and registers.
  - Multi-Core Aware.
  - Interrupt or RapidIO Door Bell generation and/or status bit indication on job or multiple jobs completion.
  - System memory utilized only for input/output data, all the internal calculations are performed using MAPLE-B memories.
— Turbo Decoding
  - Scalable architecture with 1, 2 or 4 Radix 4 dual-recursion engines.
  - Supports 3GPP-R6 turbo decoding as specified in 3GPP TS 25.212, section 4.2.3.2.
  - Supports 3GPP2 turbo decoding as specified in 3GPP2 C.S002, section 2.1.3.1.4.2.
  - Supports Wimax OFDMA turbo decoding as specified in **IEEE**® 802.16™-2004 standard and corrigendum **IEEE** P802.16-2004/Cor2/D2 standard, section 8.4.9.2.3.
  - Supports 3GLTE (Evolved UTRA) turbo decoding as specified in 3GPP TS 36.212, section 5.1.2.2.
  - Programmable Number of Iterations

- Multiple stop conditions, including built in CRC check and A-Posteriori Quality Indication.
- Programmable de-puncturing schemes including support for rate-de-matching in 3GPP (and 3G LTE).
- Binary and Duo-Binary turbo codes.
- Support for trellis termination bits and tail biting.
- Decoding techniques using: Max Log Map or Linear Log Map (MAX*) including extrinsic factorization.
- 8-bit soft symbol inputs with Hard or Soft decision outputs.

— Viterbi Decoding

- Supports 3GPP-R6 viterbi decoding as specified in 3GPP TS 25.212, section 4.2.3.1.
- Supports 3GPP2 viterbi decoding as specified in 3GPP2 C.S002, section 2.1.3.1.4.1.
- Supports Wimax OFDMA channel decoding as specified in **IEEE** 802.16-2004 and corrigendum **IEEE** P802.16-2004/Cor2/D2, section 8.4.9.2.1.
- Supports 3GLTE (Evolved UTRA) channel decoding as specified in 3GPP TS 36.212, section 5.1.2.1.
- Fully programmable polynomials
- Programmable schemes for supporting various rates/puncturing cases (e.g. 1/2, 1/3).
- Support for zero tailing.
- Support for tail-bitting with multiple-iteration or WAVA* approaches.
- 8-bit soft symbol inputs with Hard or Soft decision outputs.

— FFT/iFFT and DFT/iDFT processing

- Variable lengths FFT/iFFT processing of lengths 128, 256, 512, 1024 and 2048.
- Variable lengths DFT/iDFT processing of the form $2^k \cdot 3^m \cdot 5^n \cdot 12$, up to 1536 points.
- Mixed radix implementation using R2, R3, R4, R5 and R8 building blocks.
- 16-bit I, 16-bit Q input, output and twiddles resolutions.
- Input and output I/Q samples are interleaved in memory.
- Internal twiddles memory.

— CRC processing with the following features:

- Four possible polynomials
- CRC check or CRC calculation for block sizes of up to 128 Kb
- Optional byte reverse CRC processing
- CRC processing with throughput of up to 12 Gbps at 450 MHz

— When it is not required, the MAPLE-B power can be disabled internally to reduce overall device power consumption.

- The Security Engine (SEC) includes 8 different execution units (EUs). For EUs for which data flows in and out, each EU has buffer FIFOs of at least 256 bytes. EU types and features include the following:
  — Advanced Encryption Standard Unit (AESU)
    - Implements the Rijndael symmetric key cipher per U.S. National Institute of Standards and Technology FIPS 197.
    - Modes providing data confidentiality: ECB, CBC, CCM, Counter, GCM, XTS, CBC-RBP, OFB-128, and CFB-128.
    - Modes providing data authentication: CCM, GCM, CMAC (OMAC1), and XCBC-MAC.
    - 128, 192, 256 bit key lengths (only 128 bit keys in XCBC-MAC)
    - ICV checking in CCM, GCM, CMAC (OMAC1), and XCBC-MAC mode
    - XOR operations on 2–6 sources for RAID applications
  — ARC Four Execution Unit (AFEU)
    - Implements a stream cipher compatible with the RC4 algorithm
    - 8-bit to 128-bit programmable key
  — Cyclic Redundancy Check Unit (CRCU)
    - Implements CRC32C as required for iSCSI header and payload checksums, CRC32 as required for **IEEE** Standard 802 packets, as well as for programmable 32 bit CRC polynomials
    - ICV checking
  — Data Encryption Standard Execution Unit (DEU)
    - DES, 3DES
    - Two key (K1, K2, K1) or Three Key (K1, K2, K3)
    - ECB, CBC, CFB-64 and OFB-64 modes for both DES and 3DES
  — Kasumi Execution Unit (KEU)
    - Implements cipher and authentication modes F8 and F9 used in 3G, A5/3 for GSM and EDGE, and GEA3 for GPRS
    - 128-bit confidentiality key and 128-bit integrity key
    - ICV checking for F9
  — SNOW3G Execution Unit (STEU)
    - Implements cipher and authentication modes UEA2 (F8) and UIA2 (F9)
    - 128-bit confidentiality key and 128-bit integrity key
    - ICV checking for F9
  — Message Digest Execution Unit (MDEU)

- Implements SHA with 160-bit, 224-bit, 256-bit, 384-bit, and 512-bit message digest (as specified by the FIPS 180-2 standard)
- Implements MD5 with 128-bit message digest (as specified by RFC 1321)
- Implements HMAC computation with either message digest algorithm (as specified in RFC 2104 and FIPS-198)
- Implements SSL MAC computation
- ICV checking

— Public Key Execution Unit (PKEU)

- RSA and Diffie-Hellman with programmable field size up to 4096 bits
- Elliptic curve cryptography
  - — F_2m and F_p modes
  - — Programmable field size up to 1023 bits
- Run time equalization to protect against timing and power attacks

— Random Number Generator (RNGU). Combines a True Random Number Generator (TRNG) and a NIST-approved Pseudo-Random Number Generator (PRNG) (as described in Annex C of FIPS140-2 and ANSI X9.62).

■ Chip-level arbitration and switching system (CLASS)
— A full fabric that arbitrates between the DSP cores and other CLASS masters to the core M2 memory, shared M3 memory, DDR SDRAM controllers, MAPLE-B, and the device configuration control and status registers (CCSRs).
— High bandwidth.
— Non-blocking allows parallel accesses from multiple initiators to multiple targets.
— Fully pipelined.
— Low latency.
— Per target arbitration highly optimized to the target characteristics using prioritized round-robin arbitration.
— Reduces data flow bottlenecks and enables high-bandwidth internal data transfers.

■ Internal memory. The 4608 Kbyte internal memory space includes:
— 32 Kbyte L1 ICache per core.
— 32 Kbyte L1 DCache per core.
— 512 Kbyte unified L2 Cache / M2 Memory per core.
— 1056 Kbyte shared M3 memory. 1024 Kbyte of M3 memory can be turned off to save power, if necessary, which reduces the M3 memory size to 32 Kbyte.
— 96 Kbyte boot ROM accessible from the cores.

- Clocks
  - Three input clocks:
    - Input clock.
    - Two differential input clocks (one per each serial RapidIO PLL).
  - Five PLLs:
    - Three system PLLs
    - Two Serial RapidIO PLLs.
  - Clock ratios selected during reset via reset configuration pins.
  - Clock modes user-configurable after reset.
- Two DDR Controllers, each supporting:
  - Up to 400 MHz clock rate (800 MHz data rate).
  - Supports both DDR2 and DDR3 devices
  - Programmable timing supporting both DDR2 and DDR3 SDRAM (but not simultaneously)
  - Support for a 64-bit data interface (72 bits including ECC), up to 800 MHz data rate, for DDR2 and DDR3
  - Support for a 32-bit data interface (40 bits including ECC), up to 800 MHz data rate, for DDR2 and DDR3
  - Full ECC support for single-bit error correction and multi-bit error detection up to the maximum specified data rates for DDR2 and DDR3
  - Two banks of memory via two chip selects. Each chip select supports up to 512 Mbytes, but the sum of the memory cannot exceed 512 Mbyte total (1 Gbyte total for the two controllers).
  - DRAM chip configurations from 64 Mbits to 4 Gbits with x8/x16 data ports
  - Support burst lengths of 4 beats for DDR2 devices
  - Support burst lengths of 4 (burst chop) and 8 beats for DDR3 devices
  - Sleep mode support for self-refresh SDRAM
  - On-die termination support
  - Supports auto refreshing
  - Support for both Unbuffered and Registered DIMMs
- DMA Controller
  - 32 unidirectional channels, providing up to 16 memory-to-memory channels.
  - Buffer descriptor programing model.
  - Up to 1024 buffer descriptors per channel direction provide a total of 32 Kbyte buffer descriptors. Buffer descriptors can reside in M2 or DDR memories.
  - Priority-based time-multiplexing between channels, using four internal priority groups with round-robin arbitration between channels on equal priority group.
  - Earliest deadline first (EDF) priority scheme that assures task completion on time.
  - Flexible channel configuration with all channels supporting all features.
  - A flexible buffer configuration, including:

- Simple buffers
- Cyclic buffers
- Single address buffers (I/O device).
- Incremental address buffers
- Chained buffers
- 1D to 4D buffers, optimized for video applications
- 1D or 2–4D complex buffers, a combination of buffer types

— Two external DMA request (DREQ) and two $\overline{\text{DONE}}$ signal lines that allow an external device to trigger DMA transfers.
— High bandwidth
— Optimized for DDR SDRAM

■ High-Speed Serial Interface (HSSI)
— Serial RapidIO® Subsystem
- Two serial RapidIO ports supporting x1/x4 operation up to 3.125 Gbaud with a RapidIO messaging unit and two RapidIO DMA units.
- Each x1/x4 serial RapidIO endpoint operates at 1.25/2.5/3.125 Gbaud and complies with the following parts of Specification 1.2 of the RapidIO trade association interconnect specification:
  — Part I (input and output logical specifications)
  — Part II (message passing logical specification)
  — Part III (common transport specification)
  — Part VI (physical layer x1 LP-serial specification)
  — Part VIII (error management extension specification)
- Each serial RapidIO port supports read, write, messages, doorbells, and maintenance accesses:
  — Small and large transport information field only
  — All priorities flow
  — Pass-through between the two ports that allows cascading devices using the serial RapidIO and enabling message/data path between the two serial RapidIO ports without core intervention. A message/data that is not designated for the specific device passes through it to the next device.
- RapidIO Messaging Unit supports:
  — Two outbound message queues
  — Two inbound message queues
  — One outbound doorbell queue
  — One inbound doorbell queue

- One inbound port-write queue

- Each RapidIO DMA unit supports:
  - Four 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)
  - Programmable bandwidth control between channels
  - Up to 256 bytes for DMA sub-block transfers to maximize performance over the RapidIO interface
  - Three priority levels supported for source and destination transactions

- PCI-Express Controller
  - Complies with the *PCI Express™ Base Specification, Revision 1.0a*
  - Supports root complex (RC) and endpoint (EP) configurations
  - 32- and 64-bit address support
  - x4, x2, and x1 link support
  - Supports accesses to all PCI Express memory and I/O address spaces (requestor only)
  - Supports posting of processor-to-PCI Express and PCI Express-to-memory write
  - Supports strong and relaxed transaction ordering rules
  - 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 and one configuration window
  - Supports four outbound translation windows and one default window
  - Supports eight non-posted and four posted PCI Express transactions
  - Supports up to six priority 0 internal platform reads and eight priority 0 to 2 internal platform writes. (The maximum number of outstanding transactions at any given time is eight.)
  - Credit-based flow control management
  - Supports PCI Express messages and interrupts

- Dual x4 SerDes Ports:
  - Port 1 supports x4 serial RapidIO interface or x1 serial RapidIO interface and two SGMII ports

- Port 2 supports x1/x4 serial RapidIO interface or x1/x2/x4 PCI Express interface and two SGMII ports

- The QUICC Engine subsystem includes dual RISC processors and 48-Kbyte multi-master RAM to handle the Ethernet and SPI interfaces, thus off loading the tasks from the cores. The three communication controllers support:
  - Two Ethernet Controllers
    - Two Ethernet physical interfaces:
      - 1000 Mbps SGMII protocol using a 4-pin SerDes interface multiplexed through the HSSI SerDes port.
      - 1000 Mbps RGMII protocol
    - MAC-to-MAC connection in all modes
    - Full-duplex operations
    - Full-duplex flow control feature (**IEEE** Std. 802.3x™)
    - Receive flow control frames
    - Detection of all erroneous frames as defined by **IEEE** Std. 802.3®-2002
    - Multi-buffer data structure
    - Diagnostic modes: Internal and external loopback mode and echo mode
    - Serial management interface MDC/MDIO
    - Transmitter network management and diagnostics
    - Receiver network management and diagnostics
    - VLAN Support
    - **IEEE** Std. 802.1p/Q™ QoS
    - Eight Tx/Rx queues
    - Queuing decision for IP/MAC/UDP filtering based on MAC destination addresses, IP destination address, and UDP destination port
    - Programmable maximum frame length
    - Enhanced MIB statistics
    - Optional shift of data buffer by two bytes for L3 header alignments
    - Extended features
      - IP header checksum verification and calculation
      - Parsing of frame headers and adding a frame control block at the frame head, containing L3 and L4 information for CPU acceleration
  - Serial peripheral interface (SPI)
    - Four-signal interface (SPI_MOSI, SPI_MISO, SPI_CK and SPI_SL)
    - Full-duplex operation

- Works with 32-bit data characters, or with a range from 4-bit to 16-bit data characters•Supports back-to-back character transmission and reception
- Supports master or slave SPI mode
- Supports multiple-master environment
- Continuous transfer mode for automatic scanning of a peripheral
- Maximum clock rate is (QUICC Engine clock)/8 in master mode and (QUICC Engine clock)/4 in slave mode (not in back-to-back operation)
- Independent programmable baud rate generator
- Programmable clock phase and polarity
- Local loopback capability for testing
- Open-drain outputs support multimaster configuration
- Communication with Ethernet PHY for configuration and status (MIIMCOM-MII management communication protocol)
- Multi-MIIMCOM environment with up to 32 PHYs
- Programmable clock gap between two characters in master mode
- Controlled by the DSP cores and the QUICC Engine RISC processors according to user configuration.

■ TDM

— Backward-compatible with the MSC8102/MSC812x/MSC814x TDM interface
— All the four TDM modules together support up to 1K time-slots for receive and 1K time-slots for transmit
— Up to four independent TDM modules:

- *Independent receive and transmit mode*. Independent transmitter and receiver. Transmitter input clock, output data, and frame sync can be configured as either input or output. Up to 256 transmit channels and up to 256 receive channels. Receiver input clock, input data, and input frame sync.
- *Shared sync and clock mode*. Two receive and two transmit links share the same clock and frame sync. The sync can be configured as either input or output. Up to 128 transmit channels and 128 receive channels.
- *Shared data link*. Up to four full-duplex data links can operate as either transmit or receive. All links have the same clock and frame sync. Each link supports up to 128 channels.

— Word size of 2, 4, 8, or 16-bit. All the channels share the same size.
— Hardware A-law/µ-law conversion
— Up to 62.5 Mbps data rate for all TDM modules
— Up to 16 Mbyte per channel buffer (granularity 8 bytes), where A/µ law buffer size has double size (16-byte granularity)
— Separate or shared interrupts for receive and transmit with two programmable receive and two programmable transmit thresholds for double buffering

- Each channel can be programmed as active or inactive
- Support either 0.5 ms (4 frames) or 1 ms (8 frames) latency
- Glueless interface to E1/T1 framers

- I/O Interrupt Concentrator consolidates all chip maskable interrupt and non-maskable interrupt sources and routes them to $\overline{\text{INT\_OUT}}$, $\overline{\text{NMI\_OUT}}$, and the cores.
- UART
  - Bit rate up to 6.25 Mbps
  - Two signals for transmit data and receive data
  - Full-duplex operation
  - Standard mark/space non-return-to-zero (NRZ) format
  - 13-bit baud rate selection
  - Programmable 8-bit or 9-bit data format
  - Separately enabled transmitter and receiver
  - Programmable transmitter output polarity
  - Separate receiver and transmitter interrupt requests
  - Receiver framing error detection
  - Hardware parity checking
  - 1/16 bit-time noise detection
  - Single-wire and loop operations
- Timers
  - Two general-purpose 32-bit timers for RTOS support per SC3850 core
  - Four TMR modules, each with the following features:
    - Four 16-bit timers
    - Cascadable timers
    - Count up/down
    - Programmable count modulo
    - Count once or repeatedly
    - Counters are preload able
    - Compare registers can be preloaded
    - Counters can share available inputs
    - Separate prescaler for each counter
    - Each counter has capture and compare capability
    - Can use one of the following clock sources: system clock, TDM clock input, or external clock input
  - Eight software watchdog timer (SWT) modules
- Eight programmable hardware semaphores, locked by simple write access without need for read-modify-write operation by the DSP core.
- Virtual interrupts

- Generation of 32 virtual interrupts by a simple write access
- Generation of virtual $\overline{\text{NMI}}$ by a simple write access

- I^2C interface
  - Two-wire interface
  - Multi-master operational
  - Calling address identification interrupt
  - START and STOP signal generation/detection
  - Acknowledge bit generation/detection
  - Bus busy detection
  - Programmable clock frequency
  - On-chip filtering for spikes on the bus
- General-purpose input/output (GPIO) ports:
  - 32 GPIO ports
  - Each GPIO port can either serve the on-device peripherals or act as a programmable I/O pin
  - Sixteen GPIO pins can be configured as external interrupt inputs
  - All ports are bidirectional
  - All ports are set as GPIO inputs at system reset
  - All port values can be read while the pin is connected to an internal peripheral
  - All ports have open-drain output capability
- Boot interface options:
  - Ethernet
  - Serial RapidIO interface
  - I^2C
  - SPI
- JTAG. Test Access Port (TAP) and Boundary Scan Architecture designed to comply with **IEEE** Std. 1149.1™.
- Reduced power dissipation
  - Very low power CMOS design
  - Low-power standby modes
  - Optimized power management circuitry (instruction-dependent, peripheral-dependent, and mode-dependent)
  - Technology: The MSC8156E device is manufactured using CMOS 45 nm SOI technology.

1.2 Block Diagram

A block diagram of the MSC8156E is shown in **Figure 1-1**.

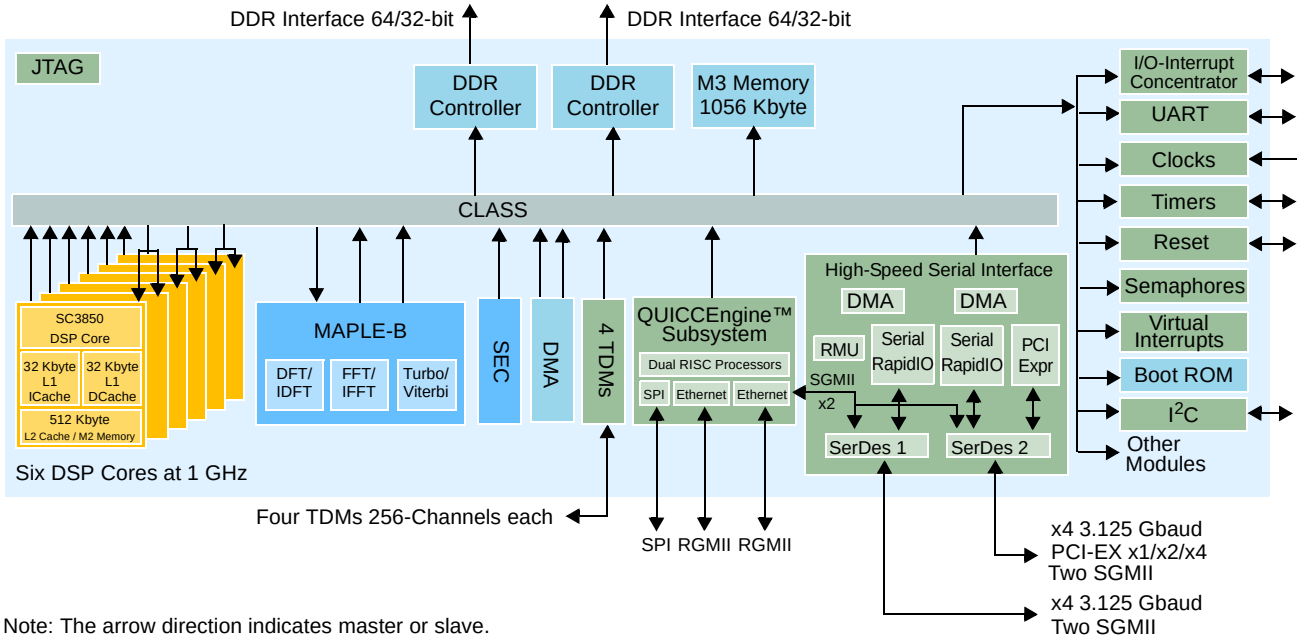


Figure 1-1. MSC8156E Block Diagram

1.3 Architecture

The MSC8156E architecture is carefully optimized to achieve the maximum channel density for a given device area, power, and cost. Also, the MSC8156E is a derivative of the same system internal platform Freescale uses to implement new DSPs. Therefore, Freescale can swiftly spin off DSP devices from the same platform and provide the customer with familiar modules and programming models.

1.4 StarCore SC3850 DSP Subsystem

Figure 1-2 shows the block diagram of the StarCore SC3850 DSP subsystem, which contains the SC3850 core, the ICache, the DCache, the MMU for task and memory protection and address translation and two write buffers. In addition, there is an interrupt controller, two timers, a debug and profiling unit, and a trace write buffer. The SC3850 core fetches instructions through a 128-bit wide program bus (P-bus), and it fetches data through two 64-bit wide data buses (Xa-bus and Xb-bus). After a brief overview of the DSP platform, this section presents a subsection on each part of the platform.

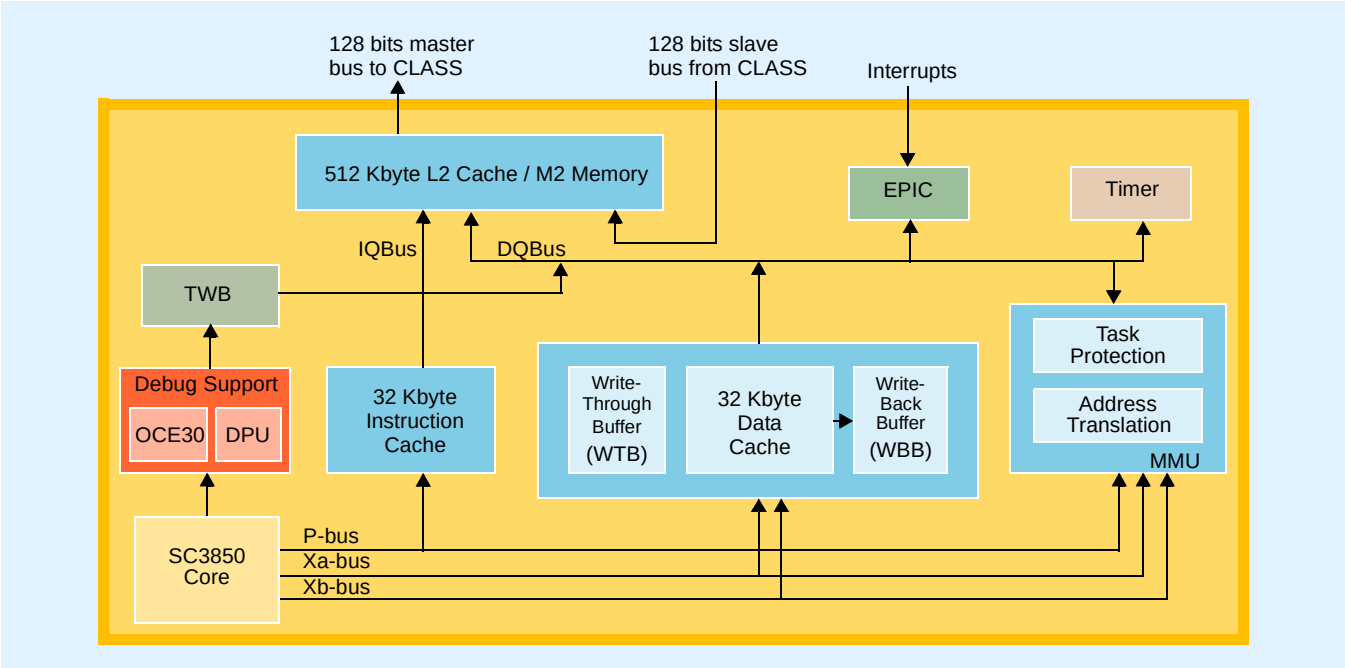


Figure 1-2. StarCore SC3850 DSP Subsystem Block Diagram

Instruction/data read accesses are performed as follows:

- Non-cacheable instructions/data are read from the target memory (for example, M2 memory).
- Cacheable instructions/data are read from the ICache/DCache. If they do not reside in the cache (a miss), they are first fetched directly from the target memory.

There are three write policies when writing data outside the core:

- *Cacheable write-back*. Information is written only to the cache. The modified cache lines are written to main memory only when they are replaced. The subsequent write-back buffer is combined with the write-allocate write-miss policy in which the required lines are loaded to the cache whenever a write-miss occurs.
- *Cacheable write-through*. Both the cache and the higher-level memory are updated during every write operation. In the StarCore SC3850 DSP subsystem, the write-through buffer is

a non-write allocate buffer. Therefore, a cacheable write-through access does not update the cache unless there is a hit.

- *Non-cacheable*. The write is direct to memory and is not written to the cache. A hazard mechanism ensures that read accesses read updated data.

The DSP subsystem supports a Real-Time Operating System (RTOS) as follows:

- Virtual-to-physical address translation in the MMU.
- Two privilege levels: user and supervisor.
- Memory protection.
- Precise exceptions upon an MMU violation enabling dynamic memory management.

The embedded programmable interrupt controller (EPIC) handles up to 256 interrupts with 32 priorities, 222 of which are external platform inputs.

1.4.1 Enhancements

Table 1-1 summarizes the major improvements and benefits of the SC3850 DSP core and subsystem. For enhancement details, see the *SC3850 DSP Core Reference Manual*.

Table 1-1. SC3850 DSP Core and Subsystem Major Benefits

No.	Feature	Type DSP/Control	Highlights	Benefit
1	Dual Multiply ALU	DSP	• Eight 16 × 16 multiplications per cycle • Complex operations • Mixed/Double precision multiplications support	Double the throughput of convolution based kernels, complex arithmetics, and mixed/double precision multiplications
3	Condition Handling instruction set architecture	Control	• Parallel condition computation	Accelerate decision making in control code
4	Microarchitecture core stall reduction	Both	• HW loop stall removal • BTB enlargement • Deeper speculation depth • Misc. stall reductions	Significant control cycle improvements and a friendlier compiler target
5	Private L2 cache	Both	• Low latency • Unified program and data	Significant out-of-the-box improvement
6	Cache instruction set architecture	Both	• Cache fetch • Cache flush/sync • Cache allocation	Significant increase in core utilization Simplified cache coherency management
7	Microarchitecture cache stall reduction	Both	• Write stall hiding • Read contention buffer	Significant increase in core utilization

1.4.2 StarCore SC3850 DSP Core

The SC3850 core is a flexible, programmable DSP core that handles compute-intensive communications applications, providing high performance, low power, and high code density. It is fully binary-backward compatible with the MSC8101, MSC8102, MSC8103, MSC8112, MSC8113, MSC8122, MSC8126, MSC8144, and MSC8144E DSPs, and it introduces many new features and enhancements.

The SC3850 core includes a data arithmetic logic unit (DALU) that contains four arithmetic logic units (ALUs). The core also includes an address generation unit (AGU) that contains two address arithmetic units. The SC3850 efficiently deploys the variable-length execution set (VLES) execution model, allowing grouping of up to 4 DALU and 2 AGU instructions in a single clock cycle without sacrificing code size for unused execution slots.

Each ALU has two 16-bit × 16-bit multipliers and a 40-bit accumulation capability, a 40-bit parallel barrel shifter and a 40-bit adder/subtractor. Each ALU performs one MAC operation per clock cycle, so a single core running at 1 GHz can perform up to 8 GMACS. Each AAU in the AGU can perform one address calculation and drive one data memory access per cycle. Data access widths are flexible from 8 to 64 bits. The AGU can support a throughput of up to 128 Gbps between the core and the memory.

Arithmetic operations use both fractional and integer data types, enabling the user to choose an individual style of code development or to use coding techniques derived from an application-specific standard. Parts of many algorithms use data with reduced width such as 8 or 16 bits. For better efficiency, the SC3850 core also supports single-instruction multiple-data (SIMD) instructions working on 2-word or 4-byte operands packed in a register. This packing allows the core to perform 2 to 4 operations per instruction (a maximum of 10 to 18 operation per VLES including AGU operations).

A new dual 20-bit packed data format enables you to accumulate two multiplication results from the dual multiply ALU into a single register with guard bits. Alternatively, accumulation of both multiplies can be combined into a single 40-bit accumulator (dot product). In addition, the SC3850 supports special instructions to support special operations, such as Viterbi and video applications.

Although the SC3850 is a DSP, the rich instruction set also gives special attention to control code, making the SC3850 core ideal for applications that embed DSP and communications operations as general control code. Among the features that support control code are the interlocked pipeline that solves dependency hazards. The powerful SC3850 compiler translates code written in C/C++ into parallel fetch sets and maintains high code density and/or high performance by taking advantage of these features and the compiler-friendly instruction set. Even compiled pure control code yields results with high code density.

The SC3850 core supports general micro-controller capabilities, making it a suitable target for advanced operating systems. These capabilities include support for user and supervisor privilege levels that enable (with the off-core MMU) a protected software model implementation. Precise exceptions for memory accesses allow implementation of advanced memory management schemes and soft error correction.

The SC3850 core includes a dynamic branch prediction mechanism that contains a 48-entry branch target buffer (BTB) to improve performance by reducing the change of flow latency.

1.4.3 L1 Instruction Cache

The instruction channel, which comprises the instruction cache (ICache) and the instruction fetch unit (IFU), provides the core with instructions that are stored in higher-level memory. The ICache operates at core speed and stores recently accessed instructions. Whenever an addressed instruction (from the cacheable memory area) is found in the array, it is immediately made available to the core (ICache hit). When the required address is not found in the array, it is loaded to the ICache from the external (off-subsystem) memory by the IFU (ICache miss). The IFU operates in parallel with the core to implement a HW line prefetching algorithm that loads the ICache with information that has a high probability of being needed soon. This action reduces the number of cache misses. When an instruction is addressed from a non-cacheable area, the IFU fetches it directly to the XP bus of the core without writing it to the cache.

1.4.4 L1 Data Cache

The data channel comprises the data cache (DCache), the data fetch unit (DFU), the data control unit (DCU), the write-back buffer (WBB), and the write-through buffer (WTB). This two-way channel reads and writes information from the core to/from higher-level memory (M2 or L2) and control memory (internal blocks and external peripherals) spaces.

The DCache, which operates at core speed, keeps the recently accessed data. When addressed data (from a cacheable memory area) is found in the array, it is immediately made available to the core (DCache hit) in a read and updated if written to. When the required address is not found in the array, a DCache miss occurs, and the DFU loads the data to the DCache from the external (off-subsystem) memory and drives it to the core. The DFU operates in parallel with the core and implements a HW line prefetch algorithm that loads the DCache with information that has a high probability of being needed soon, thus reducing the number of data cache misses.

The channel differentiates between cacheable and non-cacheable addresses. For cacheable addresses, it supports the write-back allocate and write-through writing policies. The selection is made on an address segment basis, as programmed in the MMU. The data channel supports the arrangement of data in big-endian formats. Core data types can be byte, word, long (4 bytes), or 2 long (8 bytes) wide.

1.4.5 L2 Unified Cache/M2 Memory

The L2 cache processes data and program accesses to the external M3/DDR memory. Caching the accesses requested by the L1 subsystem reduces the average penalty of accessing the high latency M3. The L2 cache includes a slave arbitration and tag unit, cache logic and arrays, along with a write buffer for write back and write through accesses, fetch logic to fetch data from the off platform memory upon a miss or a non-cacheable access, and a master arbiter that arbitrates between the different internal units.

1.4.6 Memory Management Unit (MMU)

The MMU performs three main functions:

- Memory hardware protection for instruction and data access with two privilege levels (user and supervisor).
- High-speed address translation from virtual to physical address to support memory relocation.
- Cache and bus controls for advanced memory management

Memory protection increases the reliability of the system so that errant tasks cannot ruin the privileged state and the state of other tasks. Program and data accesses from the core can occur at either the user or supervisor level. The MMU checks each access to determine whether it matches the permissions defined for this task in the memory attributes and translation table (MATT). If it does not, the access is killed and a memory exception is generated.

1.4.7 Debug and Profiling Unit (DPU)

The on-chip emulator (OCE) and the debug and profiling unit (DPU) are hardware blocks for debugging and profiling. The OCE performs the following tasks:

- Communicates with the host debugger through the SoC JTAG test access port (TAP) controller
- Enables the SC3850 core to enter the debug processing state upon a varied set of conditions to:
  - Single step
  - Execute core commands inserted from the host debugger to upload and download memory and core registers.
- Sets up to six address-related breakpoints on either PC or a data address
- Sets a data breakpoint on a data value, optionally combined with a data address
- Generates the PC tracing flow, optionally filtered to a subset of events such as only jumps/returns from subroutine, interrupts, and so on.

The DPU has the following characteristics:

- Enables parallel counting of subsystem events in six dedicated counters, from more than 40 events
- Filters, processes, and adds task ID and profiling information on the OCE PC trace information

1.4.8 Extended Programmable Interrupt Controller

The internal extended programmable interrupt controller (EPIC) manages internal and external interrupts. The EPIC handles up to 256 interrupts, 222 of which are external subsystem inputs. The rest of the interrupts serve internal subsystem conditions. The external interrupts can be configured as either maskable interrupts or non-maskable interrupts (NMIs). The EPIC can handle 33 levels of interrupt priorities, of which 32 levels are maskable at the core and 1 level is NMI.

1.4.9 Timer

The timer block includes two 32-bit general-purpose counters with pre-loading capability. It counts clocks at the core frequency. It is intended mainly for operating system use.

1.5 MAPLE-B

The MAPLE-B is a baseband algorithm accelerator for Turbo, Viterbi, FFT/iFFT, and DFT/iDFT algorithms The MAPLE-B consists of a programmable-system-interface (PSIF) that is a programmable controller with DMA capabilities and three accelerators attached to it using an internal interface:

- Turbo/Viterbi Processing-Element (TVPE)
- FFT Processing-Element (FFTPE)
- DFT Processing-Element (DFTPE)
- CRC Processing-Element (CRCPE)

The PSIF has two 64-bit wide MBus master ports used to transfer input and output data to and from system memory and a 64-bit MBus Slave port that allows any core to access its internal memories. Through this port, the cores can perform any of the following functions:

- Place buffer-descriptors in the PSIF internal memory.
- Access the MAPLE-B parameter RAM.
- Access the PSIF internal registers using the SBus.
- Access the process element (PE) configuration registers.

1.6 Security Engine (SEC)

The optional Security Coprocessor version 3.1.0 (SEC) performs computationally intensive security functions including the following:

- Key generation and exchange
- Authentication
- Bulk encryption.

It is optimized to process all the algorithms associated with internet protocol security (IPSec), internet key exchange (IKE), secure sockets layer/transport layer security (SSL/TLS), internet small computer system interface (iSCSI), secure real-time transport protocol (SRTP), the **IEEE** 802.11i™ security standard, worldwide interoperability for microwave access (WiMAX), third generation (G3) A3/5 for global system for mobile communication (GSM) and Enhanced Data Rates for GSM evolution (EDGE), and GEA3 for general packet radio service (GPRS). For applications requiring security protection for sensitive data, the SEC provides encryption/decryption capability without imposing process loading on the device core processors. The SEC includes a controller, four data channels, and eight execution units (EUs) including a shared random number generator (RNGU) that use a common interface to the controller. The EUs perform the specific mathematical manipulations required by protocols used in cryptographic processing.

1.7 Chip-Level Arbitration and Switching System (CLASS)

The CLASS is the central internal interconnect system for the MSC8156E device. The CLASS is a non-blocking, full-fabric interconnect that allows any initiator to access any target in parallel with another initiator-target couple. The CLASS uses a fully pipelined low latency design. The CLASS demonstrates per-target prioritized round-robin arbitration, highly optimized to the target characteristics. The CLASS operates at 500 MHz, and is separate from the SC3850 core frequency to provide an optimized trade-off between power dissipation, memory technology, and miss latency. Controlling the intradevice data flow, the CLASS reduces bottle necks and permits high bandwidth fully pipe-lined traffic.

The CLASS initiators are:

- Six SC3850 DSP subsystems
- Two MAPLE-B ports
- HSSI
- Peripheral group (TDM, QUICC Engine subsystem, SEC, and JTAG)
- Two DMA ports

The CLASS targets are:

- Configuration control and status registers (CCSRs)
- MAPLE-B
- Three core ports (each shared by two cores)
- Two DDR controllers
- M3 memory

1.8 M3 Memory

The 1056 KB M3 memory can be used for both program and data and eliminates the need for an external memory in a variety of applications, thus reducing board space, power dissipation, and cost. The M3 memory has a 128-bit wide port and runs at 500 MHz using dense memory technology. The M3 memory supports partial, full, and burst accesses. The M3 memory includes hidden refresh with a low probability of conflict with core accesses, and it supports burstable accesses. If the full M3 memory is not required, power can be turned off to 1024 MB to reduce power consumption.

1.9 Clocks

The MSC8156E device has three input clocks:

- A shared input clock.
- Two differential input clock for HSSI dual-SerDes port interface.

The MSC8156E device includes five PLLs:

- Three system PLLs to support the different internal system clock requirements required by the peripherals and interfaces.
- Two SerDes PLLs.

The ratios between the system clocks are selected during reset via reset configuration pins. The clock ratios are selected from a fixed table called clock modes table. The clock modes can be changed by the user after reset.

1.10 DDR Controllers (DDRC1 and DDRC2)

The DDR SDRAM interface is useful when the channel storage size is relatively big (as for a modem) and also when more channels are required to supplement the internal memory. When the MSC8156E device works with channel data stored in the DDR SDRAM, the DMA controller can swap the data to and from the M2 memory, thus enabling the L1 DCache to fetch from M2 memory instead of accessing the DDR SDRAM memory directly. Fetch latency is thus reduced, significantly improving the average clock cycles required per task. The M2 and M3 memories are large enough to accommodate the number of channels processed by the DSP subsystems for a variety of packet telephony and wireless transcoding application, such as basic G.711 voice

coding, G.729 or G.723 premium voice coding, AMR, and EFR. However, it is not large enough for such memory-consuming applications as a V.90 modem, especially when high channel densities are required. For these applications, the MSC8156E can interface with JEDEC-compliant DDR2 or DDR3 SDRAM devices. A DDR SDRAM can be used not only as an extension for the M2 and M3 memories but also to store code. In a typical application, infrequently used code is either swapped into M2/M3 memory when needed or executed directly from an external DDR SDRAM. The DDR SDRAM interface frequency is decoupled from the DSP subsystem frequency, and it has a separate PLL to deliver the required frequency according to the bandwidth requirements. It is 16/32 bits wide and can interface with up to two 16-bit wide devices, or four 8-bit wide devices. It has a separate strobe per byte. Two logical banks (chip select) are supported, each with logical programmable bank start and end addresses. A bank size of up to 128 MB is supported. Programmable parameters allow for a variety of SDRAM organizations and timings. Using data mask bits, the SDRAM controller enables partial write operations to bytes in a word or words in a burst. Optional ECC protection is provided for the DDR SDRAM data bus. Using ECC, the memory controller detects all two-bit errors and corrects all single-bit errors within the 32-bit data bus. For ECC, an additional ECC DDR SDRAM device is usually needed. Both the data DDR SDRAM and the ECC DDR SDRAM should have the same CAS latency. There is page retention for up to four simultaneous open pages, and the number of clocks for which the pages are kept open is programmable. Pages are replaced using a pseudo-LRU replacement algorithm.

1.11 DMA Controller

The DMA controller enables data movement and rearrangement while the DSP cores work independently. The DMA controller transfers blocks of data to and from the M2 memory, M3 memory, and the DDR SDRAM controller. It has 16 high-speed bidirectional channels and can be commanded from each of the DSP subsystems, as well as from an off-device initiator through the RapidIO or PCI using BDs. All channels are capable of complex data movement and advanced transaction chaining. Operations such as descriptor fetches and block transfers are initiated by each of the sixteen channels. Full duplex operation allows the DMA controller to read data from one target and store it in its internal memory while concurrently writing another buffer to another a target. This capability can be used extensively when data is read from the M3 memory and written into the M2 memory. The bidirectional DMA controller reads from one of the CLASS target ports while writing to the second one. The DMA controller supports smart arbitration algorithms such as round robin, bandwidth control, and a timer-based mechanism using an earliest deadline first (EDF) algorithm.

1.12 High Speed System Interface

The High Speed Serial Interface (HSSI) is a 8-port (two x4 SerDes PHYs) serial communications subsystem that supports the following multiplexed serial interface combinations:

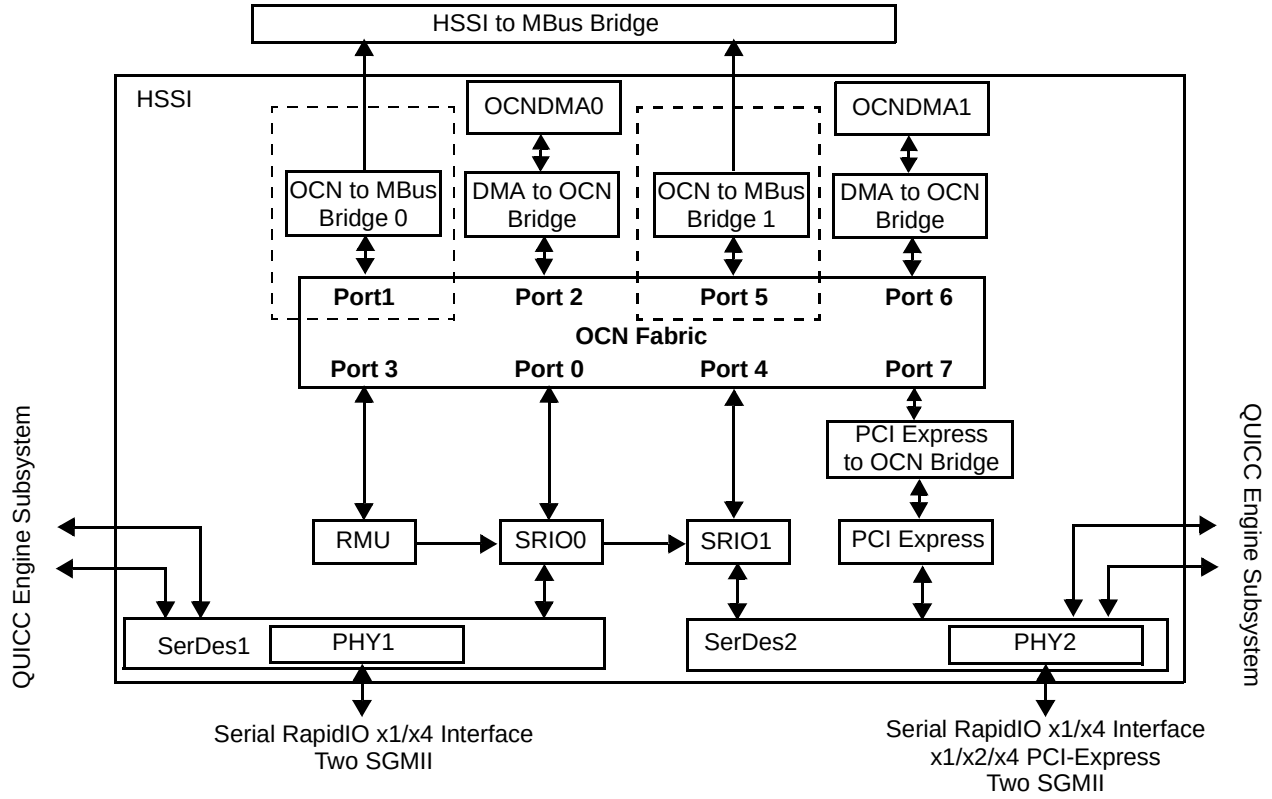
- Two x1/x4 Serial RapidIO ports
- One x1/x4 Serial RapidIO ports, one x1 Serial RapidIO port, and two SGMII ports
- One x1/x4 Serial RapidIO port and a PCI Express port
- One x1 Serial RapidIO port, two SGMII ports, and a PCI Express port

To support these interfaces, the HSSI includes the following blocks:

- One RapidIO controller with two ports and one RapidIO Messaging Unit (RMU)
- One PCI Express controller with a bridge to the OCN fabric.
- One 8-port OCN fabric with two DMA controllers to connect between the RapidIO and PCI Express controllers and the system CLASS module
- Two Serial RapidIO port controllers to link the RMU and OCN fabric to the SerDes ports.
- Two SerDes interfaces to connect to the external signal interface.

These communication interfaces allow the cores to execute the data processing code and be relieved from the data transfer and handling overhead for processing serial data flow.

Figure 1-3 shows a block diagram of the HSSI.



Notes: 1. The actual signals multiplexed for each PHY is determined by the SerDes configuration field contents in the lower 32 bits of the reset configuration word, which are recorded in RCWLR[S1P] and RCWLR[S2P]. See **Chapter 5**, *Reset* for details.

2. You must distribute the access loading between O2M0 (Port 1) and O2M1 (Port 5) for optimal performance. Although the internal OCN arbiters attempt to balance access automatically, these ports can be configured to work specifically with individual RapidIO inbound windows (see **Section 16.6.59** for details) and PCI Express inbound windows (see **Section 17.4.1.4.13** for details).

Figure 1-3. HSSI Block Diagram

1.12.1 Serial RapidIO Subsystem

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 RapidIO architecture provides a rich variety of features including high data bandwidth, low-latency capability, and support for high-performance I/O devices, as well as providing message-passing and software-managed programming models. The Serial RapidIO subsystem consists of a Serial RapidIO controller supporting two ports and a RapidIO Message Unit (RMU). The MSC8156E device can either connect directly to a host or to a Serial RapidIO switch. Each port in the switch is point-to-point connected to the MSC8156E device through a serial RapidIO link. This link typically carries packets in both directions. Packets ready for processing are transported from the host to the MSC8156E, and the processed packets are transported from the MSC8156E back to the host.

1.12.1.1 Serial RapidIO and Host Interactions

The Serial RapidIO controller directs the traffic flow between a host processor and the MSC8156E device through the RMU. The host and the MSC8156E communicate as follows:

- The host may send messages to the destination MSC8156E device, which are sent back to the host after processing along with a short doorbell interrupt to indicate that the packets have been processed.
- Messages eliminate the latency of read accesses. The host writes to the MSC8156E and the MSC8156E writes to the host. In addition, messages can be used in applications where the host does not know the internal memory structure of the target DSP.
- The host may directly access the data in the MSC8156E memory for both reads and writes. It handshakes with the software running on a DSP core through a ring of descriptors in MSC8156E memory.
- The host may access the data in MSC8156E memory for both reads and writes, but instead of maintaining a ring of descriptors in the MSC8156E memory, it uses buffer descriptors (BDs) that are messaged from the DSP core to the host.
- The host may put all the data buffers into its memory and have the MSC8156E access the data.
- Any initiator on the RapidIO system can access any internal memory space as well as the DDR SDRAM using NREAD, NWRITE, MESSAGE, and DOORBELLS. In addition, it can configure the RapidIO messaging and configuration unit using MAINTENANCE packets.

In the receive path, the following steps occur:

1. The clock is recovered using a dedicated PLL.
2. The data is deserialized, 8b/10b decoded, checked for the correct CRC, and passed to the higher-level protocol logic.
3. The control symbols are stripped and used to interface with the other peers without core intervention.
4. NWRITE packets are written to the destination memory.
5. MESSAGES are directed to the RapidIO messaging unit from which they are forwarded to their destination queue/memory location.
6. RESPONSE messages are associated with their respective NREAD and go back to the internal initiator that initiated the transaction.

On the transmit path, packets are buffered. The CRC is calculated and arbitration is performed between packet data and control symbols. The data stream then passes through the 8b/10b encoder and the serializer and transmitted on the RapidIO link. The RapidIO endpoints support link initialization and training according to the RapidIO specification. The buffers in the RapidIO endpoints support packets of up to 256 bytes and four priority levels for both the receive and the transmit.

1.12.1.2 RapidIO Messaging Unit (RMU) Operation

The messaging unit is divided into five parts:

- Inbound message controllers.
- Outbound message controllers.
- Inbound doorbell controllers.
- Outbound doorbell controller.
- Inbound maintenance controller.

The message receiver performs the following steps:

1. Filters the received packets into multiple queues (controllers) based on selected (programmable) fields in the RapidIO message header (for example, mailbox number and letter number). This filtering mechanism can be used for filtering the messages to the different SC3850 cores or filtering the messages according to their size to the right queue.
2. Writes the message to a receive buffer pre-allocated by the SC3850 core.
3. Post-increments the buffer write pointer.

4. Optionally interrupts the SC3850 core. The core can then read the buffer, process the message data, and update the read pointer of the buffer by writing it to the messaging controller.

The doorbell receiver functions in much the same way except for filtering according to a selected field in the header only.

The message transmitter performs the following steps:

1. The SC3850 core sets the registers in the controller with the message parameters (read pointer, message length, destination, buffer size, available messages.)
   - Optionally, the SC3850 core initiates only a pointer to a BD queue.
   - The messaging controller reads the BD that includes the message parameters.
2. The controller reads data from memory according to predefined parameters.
3. The controller encapsulates the message and transfer it to a RapidIO endpoint.
4. The RapidIO endpoint sends the message.
5. Acknowledges are transferred from the RapidIO endpoint to the RMU.
6. Upon completion of a message the controller can:
   - Send an interrupt to the core, waiting for a new sets of parameters.
   - Proceed to the next message in queue according to the previous parameters.
   - Proceed to the next BD in queue.

The doorbell transmitter performs the following steps:

1. The SC3850 core sets the registers in the controller with the doorbell parameters (doorbell data, destination)
2. The controller encapsulates the doorbell and transfers it to the RapidIO endpoint.
3. The RapidIO endpoint sends the message.
4. Acknowledges are transferred from the RapidIO endpoint to the RMU.
5. Upon completion of a message the controller can send an interrupt to the SC3850 core and wait for a new sets of parameters.

1.12.2 PCI Express

The PCI Express controller connects to a 2.5-GHz serial interface configurable for up to a x4 interface. As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. When selected and enabled by the device configuration, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner after it completes its reset sequence. Once link autonegotiation is successful, the controller is available to transfer 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 core processor/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.

As an initiator, the PCI Express controller supports memory read and write operations with a maximum transaction size of 256 bytes. In RC mode, the controller also supports configuration and I/O transactions. 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.

1.12.3 OCN-DMA Controllers

The MSC8156E includes two dedicated DMA controllers that transfer blocks of data between the serial RapidIO controller and the PCI Express controller and the local address space independent of the DSP cores. This offloads the data management processing from the cores. Each dedicated DMA controller offers the following features:

- Four 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
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- An Address Translation Management Unit (ATMU) with 10 local access address windows. The ATMU translates a request address into a logical device source/destination.

1.12.4 OCN Fabric

The On-Chip Network (OCN) fabric is a non-blocking high speed interconnect used for embedded system devices. The MSC8156E DSP HSSI uses an 8-port OCN to connect between the Serial RapidIO Controller, the PCI Express controller, the two supporting DMA controllers and the dual SerDes PHYs. The OCN requires no programming and provides a seamless interface for the HSSI.

1.12.5 Serial RapidIO Port Controller Modules (SRIOn)

The SRIO0 and SRIO1 modules provide an interface bridge between the RMU and the two SerDes PHYs. The units require no programming, but transfers through the SRIOn modules can be tracked by the performance monitor.

1.12.6 SerDes PHY Interfaces

The HSSI includes two 4-port SerDes interfaces that are multiplexed between the two Serial RapidIO ports, the PCI Express port, and the two SGMII ports from the QUICC Engine subsystem. Multiplexing configuration is done using the HSSI general configuration registers.

1.13 QUICC Engine Subsystem

The MSC8156E QUICC Engine module is a versatile communications engine based on a subset of the MPC83XX QUICC Engine subsystem that integrates several communications peripheral controllers. The QUICC Engine module combines interface hardware and RISC firmware to support multimedia packet operations. The QUICC Engine module includes control registers and an interrupt controller to allow the DSP cores to control and monitor operations. These registers configure certain global options and create specific commands related to the communication protocols. The cores issue commands by writing to the QUICC Engine module Command Register (QECMDR). These commands are used to initialize the RISC processors and each specific communications controller while the RISC engines are running. The QUICC Engine module includes various blocks to provide the system with an efficient way to handle data communication tasks, including:

- Two RISC processors, each of which provide:
  - One instruction per clock
  - Code execution from internal ROM or multi-port RAM
  - 32-bit RISC architecture
  - Up to sixteen internal software timers maintained in the multi-port RAM
  - Interface with the core processors through a 48-KB dual-port RAM and virtual DMA channels for each interface controller
  - Ability to handle serial protocols and virtual DMA
- Multi-initiator 48-KB multi-port RAM
- 48-KB instruction RAM (IRAM)
- Serial DMA channel
- Three full-duplex communications controllers:
  - Communications controllers 1 and 3 support IEEE 802.3/Fast Ethernet controllers
- Interrupt controller

- Multiplexer and timers logic
- Baud-rate generators

The internal clocks (RCLK/TCLK) for each communications controller can be programmed to use either an external or internal source. The rate of these clocks can be up to one-half of the QUICC Engine module clock frequency. However, the ability of an interface to support a sustained bit stream depends on the protocol settings and other factors.

Figure 1-4 shows the MSC8156E QUICC Engine module block diagram.

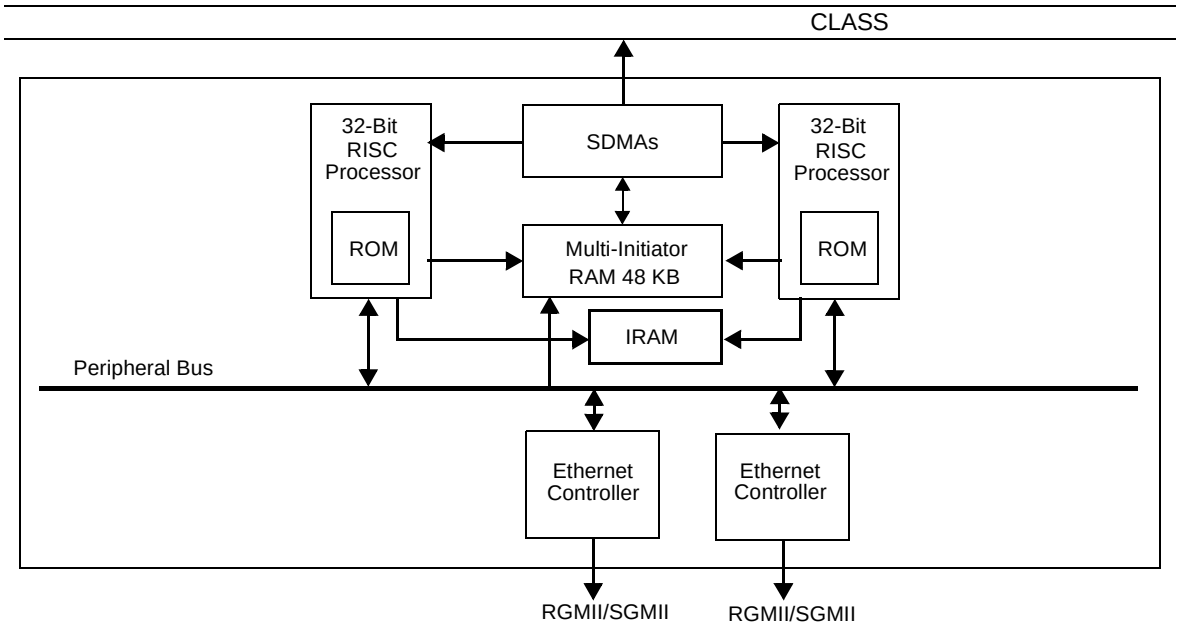


Figure 1-4. QUICC Engine Module Block Diagram

1.13.1 Ethernet Controllers

The two identical gigabit Ethernet controllers are based on the enhanced PowerQUICC II™ Ethernet controller with network statistics. The Ethernet controllers support two standard MAC-PHY interfaces to connect to an external Ethernet transceiver:

- 1000 Mbps SGMII with SerDes support
- 1000 Mbps RGMII (full duplex only)

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the Ethernet controller activates its transmit scheduler. As a result, the controller starts polling the first transmit buffer descriptor (TxBD) in one of the eight transmit queues as chosen by the scheduler. The TxBD ring is polled every 512 transmit clocks. If TxBD[R] bit is set, the Ethernet controller begins moving transmit buffers from memory to the Tx virtual FIFO. The Ethernet MAC transmitter takes data from Tx virtual FIFO and transmits the data through the appropriate interface (RGMII/SGMII) to the physical media. The transmitter, once initialized,

runs until the end-of-frame (EOF) condition is detected, unless a collision within the collision window occurs (in half-duplex mode) or an abort condition is encountered. The Ethernet Controller receiver can perform pattern matching, data extraction, Ethernet type recognition, CRC checking, VLAN detection, short frame checking, and maximum frame-length checking.

1.13.2 Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the exchange of data with other devices containing an SPI. The SPI also communicates with peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices. The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock, and slave select). The SPI block 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 SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is sent and received simultaneously.

1.14 TDM

The TDM interface connects gluelessly to common telecommunication framers, such as T1 and E1. It can interface with multiple buses, such as H-MVIP/H.110 devices, TSI, and codecs such as AC-97. It provides a total 4096 channels that are timing compliant with their clock, sync and data signals. The TDM is composed of eight identical and independent modules. Each TDM module can be configured in one of the following modes:

- *Independent receive and transmit mode*. The transmitter has an input clock, output data and a frame sync that can be configured as either input or output. There are up to 256 transmit channels and up to 256 receive channels. The receiver has an input clock, input data, and an input frame sync.
- *Shared sync and clock mode*. Two receive and two transmit links share the same clock and the frame sync: The sync can be configured as either input or output. Each of the two transmit and receive links supports up to 128 channels.
- *Shared data link mode*. Up to four full-duplex data links, which operate as either transmit or receive, have the same clock and frame sync. Each link supports up to 128 channels.

If all are used, the TDM modules can support up to 500 Mbps (250 Mbps Tx and 250 Mbps Rx) with a clock frequency up to 62.5 MHz (62.5 Mbps × 4 TDMs in each direction). Each channel can be 2, 4, 8, or 16 bits wide. All the channels share the same width during the TDM operation. When the slot size is 8 bits wide, the selected channels can be defined as A-law/μ-law. These channels are converted to 13/14 bits, which are padded into 16 bits and stored in memory. Each receive and transmit channel can be active or not. An active channel has a configurable buffer located in either in M2, M3, or DDR memory. The TDMs support either 0.5 ms (4 frames) or 1 ms (8 frames) latency. The buffers of one TDM interface are the same size and are filled/emptied at the same rate. A-law/u-law buffers are filled at twice the rate, so their buffer size is twice that

of the transparent channels. For receive, the buffers of specific TDM interface fill at the same rate and therefore share the same write pointer relative to the beginning of the buffer. When the write pointer reaches a predetermined threshold, an interrupt to the SC3850 core is generated. The SC3850 core empties the buffers while the TDM continues to fill the buffers until a second threshold line is reached and then an additional interrupt is generated to the SC3850 core. The SC3850 core empties the data between the first and the second threshold lines. Both the first and the second threshold lines are programmable. Using these threshold lines, the SC3850 core and the TDM can perform a double-buffer handshake. For transmit, the SC3850 core fills all the buffers of a TDM interface, and the TDM empties them. A similar method employing two threshold line interrupts is used for a double-buffer handshake between the SC3850 core and the TDM. You can program the interrupt as either shared for receive and transmit or separated.

1.15 Global Interrupt Controller (GIC)

The GIC receives the external and internal $\overline{\text{NMI}}$ and maskable interrupt sources and routes them to the SC3850 cores, to the $\overline{\text{INT\_OUT}}$ lines, or to the $\overline{\text{NMI\_OUT}}$ lines.

1.16 UART

The UART is used mainly for debugging. It provides a full-duplex port for serial communications by transmit data (TXD) and receive data (RXD) lines. During reception, the UART generates an interrupt request when a new character is available to the UART data register. During transmission, the UART generates an interrupt request when its data register can be written with new character. When accepting an interrupt request, an SC3850 core or external host should read the UART status register to identify the interrupt source and service it accordingly.

1.17 Timers

The MSC8156E device contains 16 identical 16-bit timers divided into four groups. Each group (TMR) contains four identical 16-bit timers, each with a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status registers, and a control register. In addition, each SC3850 subsystem includes two general purpose 32-bit timers. The MSC8156E device also includes 8 software watchdog timers. Each of the software watchdog timers can be used by any of the cores within MSC8156E as well as by an external host.

1.18 Hardware Semaphores

There are eight coded hardware semaphores. Each semaphore is an 8-bit register with a selective write protection mechanism. When the register value is zero, it is writable to any new value. When the register value is not zero, it is writable only to zero. Each SC3850 core/host/task has a unique predefined lock number (8-bit code). When trying to lock the semaphore, the SC3850

core writes its lock number to the semaphore and then reads it. If the read value equals its lock number, the semaphore belongs to that host and is essentially locked. An SC3850 core/host/task releases the semaphore by writing a 0 to it.

1.19 Virtual Interrupts

The global interrupt controller generates 26 virtual interrupts including 16 maskable interrupts, 8 VNMIs, and 2 interrupts for external interrupt and NMI outputs ($\overline{\text{INT\_OUT}}$ and $\overline{\text{NMI\_OUT}}$, respectively). A virtual interrupt/VNMI is generated via a write access to the Virtual Interrupt Generation Register (VIGR) by one of the SC3850 cores or an external host CPU.

1.20 I^2C Interface

The inter-integrated circuit (I^2C) controller enables the MSC8156E to exchange data with other I^2C devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCD displays. The I^2C controller uses a synchronous, multi-initiator bus that can connect several integrated circuits on a board. Two signals, serial data (I2C_SDA) and serial clock (I2C_SCL), carry information between the integrated circuits connected to it.

1.21 GPIOs

The MSC8156E has 32 general-purpose I/O (GPIO) ports that are multiplexed as either GPIO ports or dedicated peripheral interface ports. As GPIOs, each port is configured as an input or output (with a register for data output that is read or written at any time) by GPIO configuration registers. These registers also select the alternate functions and enable the open-drain function when ports are configured as outputs. In addition to the GPIO configuration registers, two general configuration registers (GPUER and GIER) also control the functionality when the ports are configured as inputs. The default configuration out of reset selects the primary GPIO input function with internal pull-ups. However, the ports are also disabled. GIER is used to enable the individual input ports and GPUER can disable the pull-ups for each port. Sixteen of the GPIs can also be configured as IRQ inputs. If configured as output, the GPIO ports can also be configured as open-drain (that is, configured in an active low wired-OR configuration on the board). In this mode, an output drives a zero voltage but goes to tri-state when driving a high voltage. The dedicated MSC8156E peripheral functions multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of MSC8156E applications.

1.22 Boot Options

The boot program in the internal boot ROM initializes the MSC8156E after it completes a reset sequence. The MSC8156E device can boot from an external host through the serial RapidIO interface or download a user boot program through the I^2C, SPI, or Ethernet ports.

1.23 JTAG

The dedicated user-accessible test access port (TAP) is fully compatible with **IEEE** Std. 1149.1. The MSC8156E device supports circuit-board test strategies based on this standard. For details on the standard, refer to the standard documentation.

1.24 Developer Environment

Freescale supplies a complete set of DSP development tools for the MSC8156E device. The tools provide easier and more robust ways for designers to develop optimized DSP systems. Whether the application targets a 3G-LTE, TD-SCDMA, or WiMAX system, the development environment gives the designers everything they need to exploit the advanced capabilities of the MSC8156E architecture.

1.24.1 Tools

The MSC8156E tool components include the following:

- *Integrated development environment (IDE).* Easy-to-use graphical user interface and project manager for configuring and managing multiple build configurations.
- *C compiler with in-line assembly*. The developer can generate highly optimized DSP code by exploiting the StarCore multiple-ALU architecture, with parallel fetch sets and high code density.
- *Librarian.* The developer can create application-specific DSP libraries for modularity.
- *Linker*. The developer can efficiently produce executables from object code and partition memory according to the application architecture; the linker supports code overlay.
- *Multi-Core Debugger*. Seamlessly integrated real-time, non-intrusive, multi-mode, multi-core, and multi-DSP debugger handles highly optimized DSP algorithms. The developer can choose to debug in source code, assembly code, or mixed mode. Supports RTOS-aware debugger.
- *Royalty-free RTOS.* Included with package and includes a graphical user interface (GUI) called Kernel Aware that shows task information, interrupts, and other processing elements.
- *Software Simulator*. Full chip simulation (FCS) that allows the developer to design an application and run it on the simulator before running it on the silicon. FCS is integrated under integrator developer environment (IDE), the simulator provides customers with tools to create projects and debug them as they would on silicon (high speed simultaneous transfers). In addition, there is an SC3850 subsystem performance accurate (PACC) simulator that is approximately 95% cycle accurate.
- *Profiler*. The developer can analyze and identify program design inefficiencies.

- *High Speed Run Control*. USB TAP high speed host-target interface allows users to program in Flash memory, ROM, and cache.
- *Host Platform Support*. Microsoft Windows and Solaris.
- *Development Board*. The application development system (ADS).
- *Kit for MSC8156E*. A complete system for developing and debugging real-time hardware and software.

1.24.2 Application Software

Freescale offers a broad range of DSP applications through its third-party application software partners; these applications target IP telephony, telephony modem, wireless and multimedia transcoding, and wireless base stations. Applications and software modules are listed in **Table 1-2**.

Table 1-2. Application Software Modules

Application	Modules
Baseband	WiMAX solution supporting Wave 1 features with future extension to Wave 2 and beyond.
	3G-LTE evolving kernels library.
	Optimized FFT kernels, including matrix multiplication, and so forth.
Device Drivers and Example Code	MAPLE-B driver, DMA driver, serial RapidIO driver, TDM driver, Ethernet driver, UART driver, security engine, memory allocation, and interrupt handling.
StarCore Libraries	Rich set of StarCore software libraries, including: Math (Part 1 and 2), Signal, Complex vector, Control function, Frequency domain, Filter, Common, Image Processing, Communication, and Matrix.

1.25 Example Applications

This section describes seven use cases.

1.25.1 Use Case 1: 3G-LTE Basic System

The system shown in **Figure 1-5** can be used as a 10 MHz FDD LTE system supporting 2×2 UL MIMO, 2×2 DL MIMO, 50 Mbps DL, 25 Mbps UL.

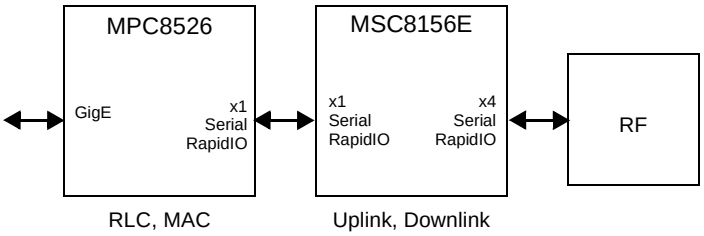


Figure 1-5. Use Case 1: 3G-LTE Basic System

1.25.2 Use Case 2: 3G-LTE System

The system shown in **Figure 1-6** can be used as an LTE 10 MHz FDD, 2 × 4 UL MIMO, 4 × 2 DL MIMO, 50 Mbit/s DL, 25 Mbps UL, 3 sectors.

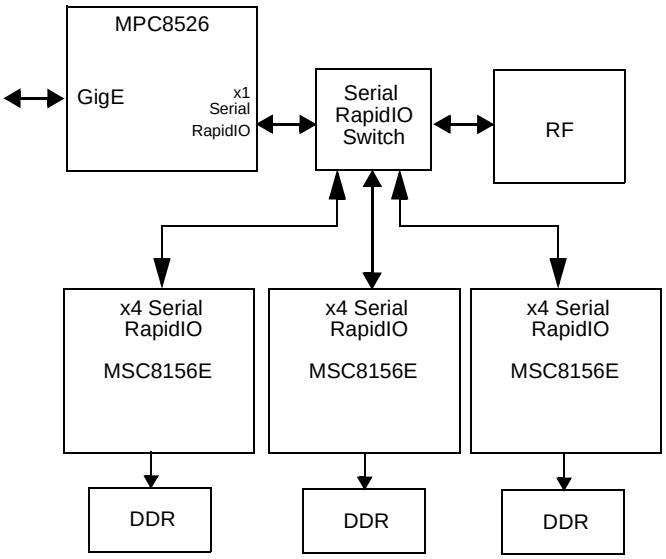


Figure 1-6. Use Case 2: 3G-LTE System

1.25.3 Use Case 3: 3G-LTE System

The system shown in **Figure 1-7** can be used as an LTE 20 MHz FDD, 2 × 2 DL MIMO, 2 × 2 UL MIMO, 100 Mbit/s DL, 50 Mbps UL, 1 sector.

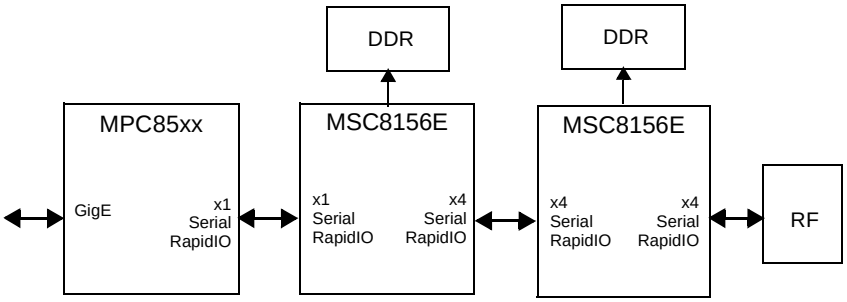
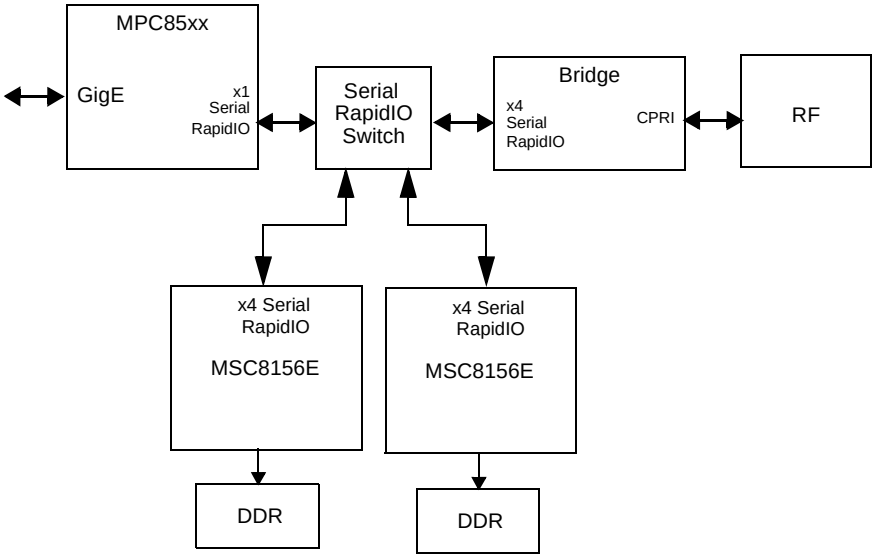


Figure 1-7. Use Case 3: 3G-LTE System

1.25.4 Use Case 4: TD-SCDMA System

The system shown in **Figure 1-8** can be used as a TD-SCDMA 1.6 MHz TDD, 8 × 8 no MIMO, 6 carriers.



Note: Two devices may be required to support 6 carriers supporting chip-rate and symbol rate, depending on the system configuration.

Figure 1-8. Use Case 4: TD-SCDMA System

1.25.5 Use Case 5: WiMAX Basic System

The system shown in **Figure 1-9** can be used as a WiMAX 10 MHz TDD, 2 × 2 MIMO, 1 sector.

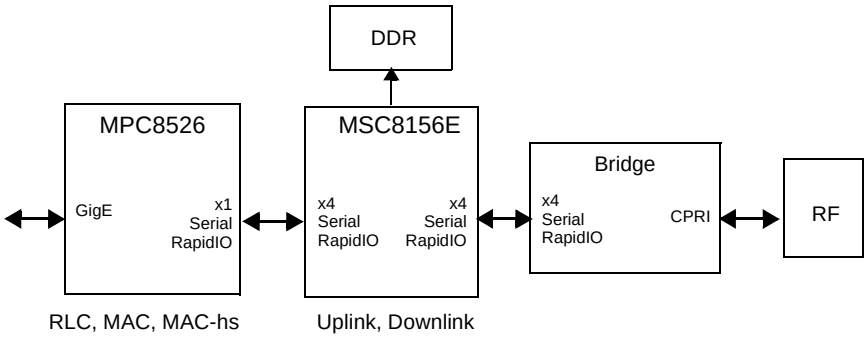


Figure 1-9. Use Case 5: WiMAX Basic System

1.25.6 Use Case 6: WiMAX System

The system shown in **Figure 1-10** can be used as a WiMAX 10 MHz TDD, 4 × 4 MIMO, 4 sectors.

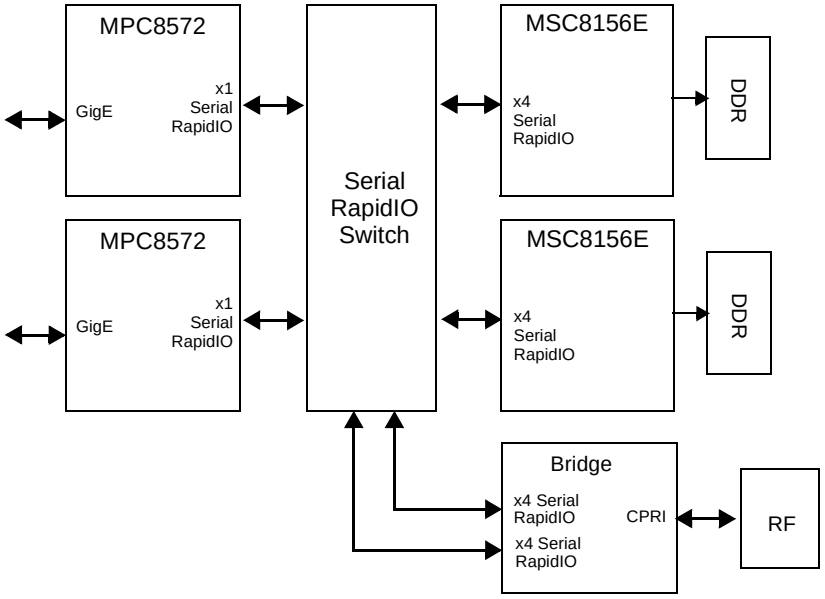


Figure 1-10. Use Case 6: WiMAX System

1.25.7 Use Case 7: WCDMA Basic System

The system shown in **Figure 1-11** can be used as a WCDMA 5 MHz FDD, 2 × 1, no MIMO, 3 sectors.

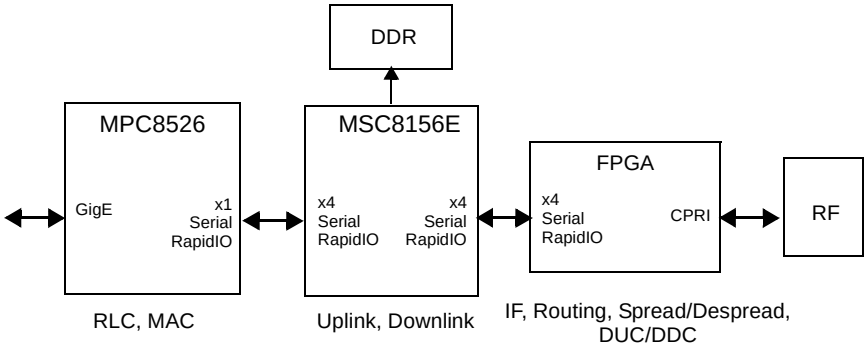


Figure 1-11. Use Case 7: WiMAX System

SC3850 Core Overview 2

The SC3850 digital signal processing (DSP) core features an innovative architecture that addresses the key market needs of DSP applications, especially in the fields of wireline and wireless infrastructure, subscriber communication, and multimedia packet transfer. This flexible DSP core supports compute-intensive applications by providing high performance, low power, efficient compile, and high code density. Each high-performance core is binary compatible with the SC140 core used in the MSC81xx DSP family, the SC1400 core, and the SC3400 core used in the MSC8144 family and delivers up to 8000 16-bit MMACS using an internal 1 GHz clock at 1 V. A MAC operation includes a multiply-accumulate command with the associated data moves and a pointer update.

The StarCore SC3850 DSP core and subsystem is an evolution of the StarCore® SC3400 DSP core and subsystem that enhances many of the original core and subsystem components and optimizes overall performance and memory hierarchy to target future application needs. Optimizations target:

- Improved control/compiled code performance
- Better operation of DSP intensive kernels
- Minimizing memory system stalls to increase core use.

Note: See the *SC3850 DSP Core Reference Manual* for a detailed description of core functionality and instruction set. The manual is only available with a signed non-disclosure agreement. Contact your local Freescale sales office or representative for details.

2.1 Core Architecture Features

Key features of the SC3850 DSP core include the following:

- Main core resources
  - 4 Data ALU execution units
  - 2 integer and address generation units
  - Sixteen 40-bit data registers with 8 guard bits, freely accessible by Data ALU instructions
  - Sixteen 32-bit address registers, freely accessible by Address generation instructions
- Instruction set
  - 16-bit instruction set, expandable to 32 and 48 instructions
  - High orthogonality of operands
  - Rich instruction set for DSP and control features
  - A very good compiler target
- Very high execution parallelism
  - Up to six instructions executed in a single clock cycle, statically scheduled
  - Variable Length Execution Set (VLES) execution model
  - Up to 4 Data ALU instructions and 2 Memory access/integer instructions per cycle
- Data type support
  - Byte (8-bit), word (16-bit) and long (32-bit) data widths, supported by instructions and memory moves
  - Both Integer (signed and unsigned) and fractional data types
  - Packed fractional complex data type
  - Several packed data types (2 to 4 objects on the same register) for SIMD operations
- Very high numerical throughput for DSP operations
  - Each Data ALU can perform two 16x16 multiplications per cycle (total of 8 multiplications for all ALUs), which can be used for:
    - Dot product acceleration (40 + (16x16) + (16x16))
    - SIMD2 multiplication and accumulation into two 20-bit register portions
    - Acceleration of Complex multiplication
    - Acceleration of extended precision multiplication
  - Some performance summary metrics are listed in **Table 2-1**:

Table 2-1. Multiplication Throughput Summary Figures for the SC3850

Operation	Precision	Instructions per Operation	Result Throughput (4 ALUs)
Real multiply	16 × 16	0.5	8
	16 × 32	1	4
	32 × 32	2	2
Complex multiply	16 × 16	2	2
	16 × 32	4	1

- Application specific instructions for acceleration the following algorithms
  - FFT
  - Video processing
  - Viterbi
  - Baseband operations
- High throughput memory interface
  - Unified, 32-bit byte addressable memory space
  - Dual Harvard architecture that permits one 128-bit program access and two 64-bit data accesses per cycle
  - Core to data memory throughput of up to 16 Giga Bytes per second, at 1 GHz core frequency
  - Support of Big Endian, Little Endian and Mixed endian memory policies
- Powerful address generation model
  - Zero overhead modulo arithmetic support for address pointers
  - Several
- Advanced pipeline
  - 12 stage, fully interlocked pipeline
  - No stalls for memory load to register, MAC operation, and result storage to memory
  - Speculation of conditionally executed instructions and change of flow execution paths
- Control features
  - Zero-overhead hardware loops with up to four levels of nesting
  - A Branch Target Buffer (BTB) for accelerating execution of change of flow instructions
- OS support
  - Precise memory exception support, for advanced OS
  - User and Supervisor privilege levels, supporting a protected, task oriented execution model
  - Full support for memory protection and address translation in the off-core MMU
  - Exception and Normal stack pointer for software stack support
  - Low task switch overhead using wide stack save and restore instructions
- Rich set of real-time debug capabilities through an On-Chip Emulator (OCE)
  - Real-time PC, data address and data breakpoint capabilities
  - Up to six hardware breakpoint channels, and unlimited debugger-enabled SW breakpoints
  - Single stepping
  - Externally forced instructions in debug mode by the host processor
  - Precise detection of PC breakpoints
  - PC tracing with filtering and compression options
  - Support for Nexus IEEE-ISTO 5001-2003 standard with off-core ready modules
- Low Power Design
  - Low-power Wait and Stop instructions
  - A very low power design
  - Fully static logic

2.2 StarCore SC3850 Core Architecture

The SC3850 core contains a Data Arithmetic and Logic Unit (DALU) with four ALUs, and an Address Generation Unit (AGU) that includes two Address Arithmetic Units (AAU). The SC3850 efficiently deploys the variable-length execution set (VLES) execution model, allowing to group up to 4 DALU and 2 AGU instructions in a single clock cycle without sacrificing code size for execution slots that are not used. **Figure 2-1** shows a block diagram of the SC3850 DSP core.

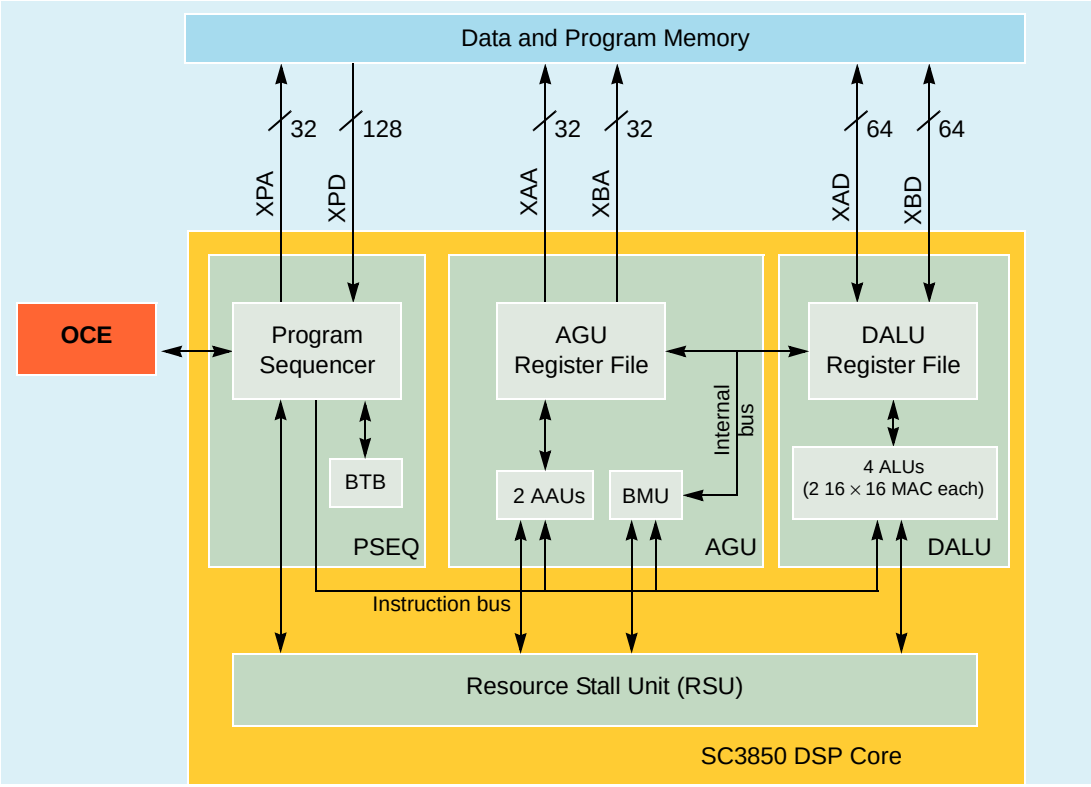


Figure 2-1. SC3850 DSP Core Block Diagram

The SC3850 uses dual-multiply ALUs supporting two 16-bit × 16-bit multipliers that can accumulate results into 40-bit wide destination data registers. In addition, it has a 40-bit parallel barrel shifter. Each ALU performs two MAC operations per clock cycle, so that a single core running at up to 1 GHz can perform 8 billion multiply-accumulates per second (GMACS). This rate is for both 16-bit operands, 8-bit operands, or mixed 8-bit and 16-bit operands.

Each AAU in the AGU can perform one address calculation and drive one data memory access per cycle. Data access widths are flexible and can be between 8 to 64 bits wide. The AGU can support a throughput of up to 128 Gbps between the core and the memory.

The program sequencer manages the instruction fetching from the program memory, dispatching the VLES to the execution units, performing change of flow (COF) and HW loop management

and exception processing. It includes a 48-entry branch target buffer which is used to accelerate the execution of COF operation.

The Resource Stall Unit (RSU) detects dependency hazards and resource requirements as defined by the code semantics. It then activates the internal operand bypass logic or inserts stalls as needed.

The SC3850 supports general microcontroller capabilities that make it a suitable target for advanced operating systems, including support for user and supervisor privilege levels that, with the MMU, enable implementation of a protected software model. The SC3850 supports precise exceptions for program and data accesses, allowing designs to implement advanced memory management schemes and soft error correction. The SC3850 also includes a 48-entry Branch Target Buffer (BTB) that improves performance by reducing the change of flow latency.

External Signals 3

The MSC8156E external signals are organized into functional groups. **Table 3-1** lists the functional groups and references the table that gives a detailed listing of signals within each group.

Table 3-1. MSC8156E Functional Signal Groupings

Functional Group	Detailed Description
Power and ground	**Table 3-3** on **page 3-4**
Clock	**Table 3-4** on **page 3-6**
Reset and Configuration	**Table 3-5** on **page 3-6**
DDR Memory Controllers	**Table 3-6** on **page 3-10**
SerDes Multiplexers (Serial RapidIO controllers, PCI Express interface, and SGMII signals)	**Table 3-7** on **page 3-11**
TDM and Ethernet	**Table 3-8** on **page 3-16**
Serial peripheral interface (SPI)	**Table 3-9** on **page 3-20**
GPIOs and maskable Interrupts	**Table 3-10** on **page 3-20**
Timers	**Table 3-11** on **page 3-25**
UART	**Table 3-12** on **page 3-26**
I^2C	**Table 3-13** on **page 3-26**
External DMA Interface	**Table 3-14** on **page 3-27**
$\overline{\text{NMI}}$/$\overline{\text{INT\_OUT}}$/$\overline{\text{NMI\_OUT}}$	**Table 3-15** on **page 3-28**
OCE module and JTAG Test Access Port	**Table 3-16** on **page 3-29**

Some signals are only sampled during the power-on reset sequence; most of these signal lines are are used by other modules and subsystems during normal operation. Signal multiplexing is determined at the following three levels:

- I/O Multiplexing. TDM channels and RGMII channels sharing. Selected during power-on reset by the GE1/GE2 bits in the high part of the Reset Configuration Word (see **Chapter 5**, *Reset* for details). Ethernet selections (RGMII/SGMII) are configured by the Ethernet registers (see the *QUICC Engine Block Reference Manual with Protocol Interworking* for details).
- SerDes Multiplexing. Serial RapidIO interfaces, PCI Express interface, and SGMII sharing on the 8-lane SerDes interface. Configured by the Reset Configuration Word (see **Chapter 5**, *Reset* for details).

- GPIO Multiplexing. GPIOs, $\overline{\text{IRQs}}$, I^2C, SPI, DMA external request interface, Timers, and UART sharing. Configured by GPIO configuration registers (see **Chapter 22**, *GPIO* for details).

The values of the GE1/GE2 RCW bits determine the sharing of signal lines between the TDM interfaces and the RGMII signals for Ethernet controllers 1 and 2. **Table 3-2** lists the signal groups supported by each of the available multiplexing modes.

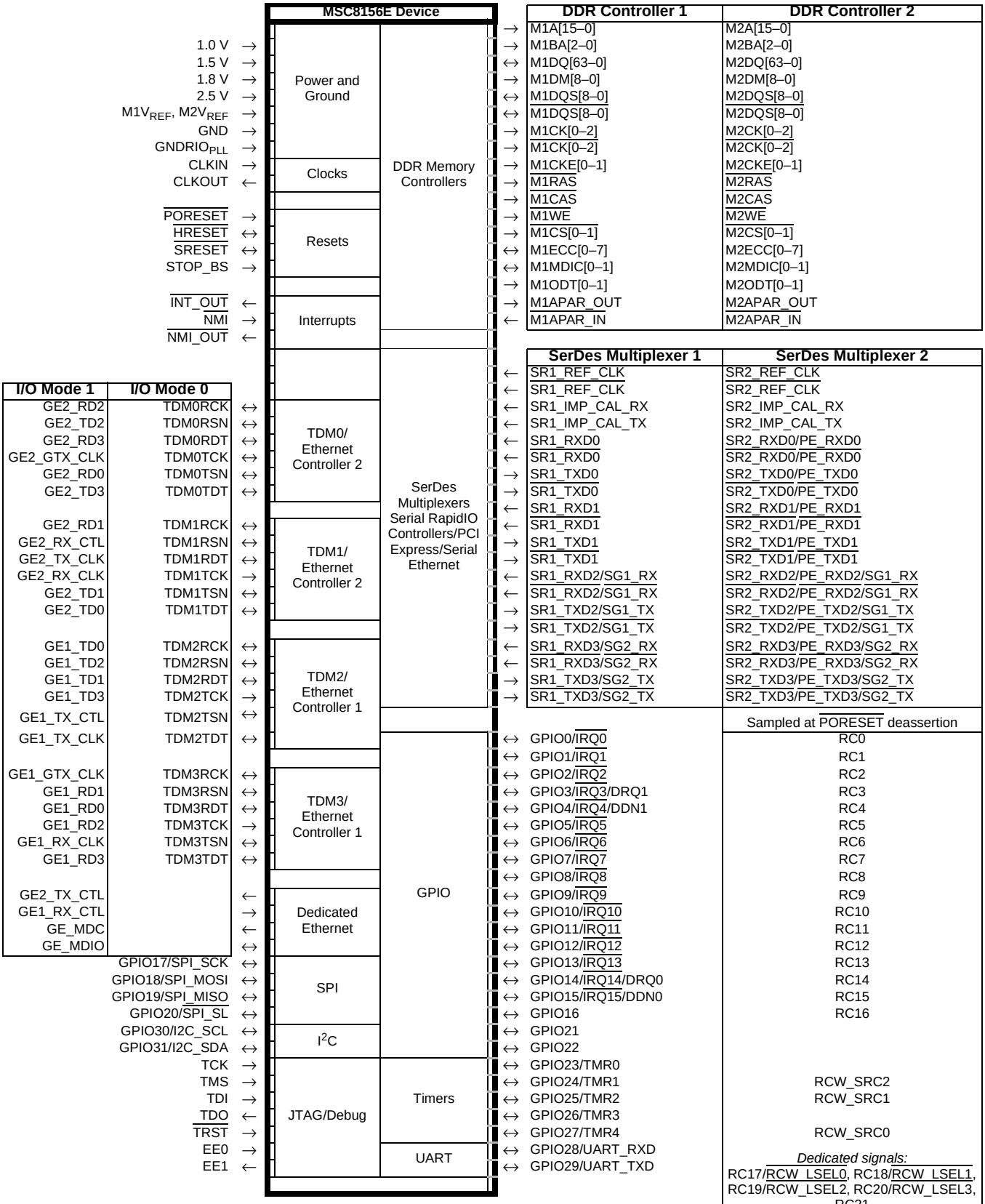
Table 3-2. Ethernet/TDM Multiplexing by GE1/GE2 Bit Values

RCW[GE1] Value	RCW[GE2] Value	Interfaces					
		TDM0	TDM1	TDM2	TDM3	GE1 RGMII	GE2 RGMII
0	0	Available	Available	Available	Available	Unavailable	Unavailable
0	1	Unavailable	Unavailable	Available	Available	Unavailable	Available
1	0	Available	Available	Unavailable	Unavailable	Available	Unavailable
1	1	Unavailable	Unavailable	Unavailable	Unavailable	Available	Available

The PCI Express, serial RapidIO interfaces, and the Ethernet SGMIIs share the two SerDes interfaces. Access to the SerDes interface blocks is configured via the RCW bits (see **Chapter 5**, *Reset* for details).

The thirty-two GPIO ports have configurable functionality. Sixteen of the GPIO lines can be configured as $\overline{\text{IRQ}}$ inputs through the GPIO configuration registers; four of these lines can also be configured as the DMA external interface. Fifteen of the other sixteen GPIO lines are multiplexed with other interface units including the SPI, timers, UART, and I^2C signals. The specific function is selected through configuration of the GPIO registers (see **Chapter 22**, *GPIO* for details).

Figure 3-1 summarizes the various MSC8156E external signal multiplexing options.



Note: See the individual signal description tables for details about individual power supplies and grounds.

Figure 3-1. MSC8156E External Signals

3.1 Power Signals

Table 3-3. Power and Ground Inputs

Nominal Voltage	Signal Name	Symbol	Description
1.0 V	VDD	V_{DD}	**Power for Core Subsystems 0–5** A dedicated well-regulated power source for core subsystems 0–5. Provide an extremely low impedance path to the V_{DD} power rail and adequate external decoupling capacitors.
	M3VDD	V_{DDM3}	**Power for M3 Memory** A dedicated well-regulated power source for 1 Mbyte of the M3 memory. Provide an extremely low impedance path to the V_{DD} power rail and adequate external decoupling capacitors. This input can be disabled when not used to reduce system power requirements. When this input is disabled, 32 Kbyte of M3 memory remains active.
	MVDD	V_{DDM}	**Power for MAPLE-B Subsystem** A dedicated well-regulated power source for the MAPLE-B subsystem. Provide an extremely low impedance path to the V_{DD} power rail and adequate external decoupling capacitors. This input can be disabled when not used to reduce system power requirements.
	PLL0_AVDD	V_{DDPLL1}	**System PLL 0 Power** A dedicated well-regulated power for the system Phase Lock Loops (PLLs).
	PLL1_AVDD	V_{DDPLL1}	**System PLL 1 Power** A dedicated well-regulated power for the system Phase Lock Loops (PLLs).
	PLL2_AVDD	V_{DDPLL1}	**System PLL 2 Power** A dedicated well-regulated power for the system Phase Lock Loops (PLLs).
1.0 V (Differential)	SXPVDD1	V_{DDSXP1}	**Serial RapidIO Interface 1 Pad Power** A dedicated well-regulated power for the Serial RapidIO interface 1 pad circuitry.
	SXPVDD2	V_{DDSXP2}	**Serial RapidIO Interface 2 Pad Power** A dedicated well-regulated power for the Serial RapidIO interface 2 pad circuitry.
	SXCVDD1	V_{DDSXC1}	**Serial RapidIO Interface 1 Core Power** A dedicated well-regulated power for the Serial RapidIO interface 1 core circuitry.
	SXCVDD2	V_{DDSXC2}	**Serial RapidIO Interface 2 Core Power** A dedicated well-regulated power for the Serial RapidIO interface 2 core circuitry.
	SR1_PLL_AVDD	V_{DDPLL1}	**Serial RapidIO PLL 1 Power** A dedicated well-regulated power for the system Phase Lock Loops (PLLs).
	SR2_PLL_AVDD	V_{DDPLL1}	**Serial RapidIO PLL 2 Power** A dedicated well-regulated power for the system Phase Lock Loops (PLLs).

Table 3-3. Power and Ground Inputs (Continued)

Nominal Voltage	Signal Name	Symbol	Description
1.5 V or 1.8 V	GVDD1	V_{DDDDR1}	**SSTL IO Driver Power (1.5 V or 1.8 V)** A dedicated power source for the DDR controller 1 DRAM interface buffers. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8156E Technical Data Sheet*.
	GVDD2	V_{DDDDR2}	**SSTL IO Driver Power (1.5 V or 1.8 V)** A dedicated power source for the DDR controller 2 DRAM interface buffers. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8156E Technical Data Sheet*.
GVDD1 × 0.5 V	M1VREF	$M1V_{REF}$	**SSTL Reference Power** A reference power level for the DDR controller 1 memory interface.
GVDD2 × 0.5 V	M2VREF	$M2V_{REF}$	**SSTL Reference Power** A reference power level for the DDR controller 2 memory interface.
2.5 V	NVDD	V_{DDIO}	**Input/Output Power** The power source for the external I/O signal lines. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8156E Technical Data Sheet*.
	QVDD	V_{DDPLL0}	**Input/Output Power** The power source for the external clock, reset, OCE, and JTAG signal lines. Provide adequate external decoupling capacitors. The external decoupling capacitors recommendations are listed in the *MSC8156E Technical Data Sheet*.
0 V (GND)	VSS	GND	**System Ground** An isolated ground for the internal processing logic and I/O buffers. This connection must be tied externally to all chip ground connections, except GND_{SXC} and GND_{SXP}.
	SR1_PLL_AGND	$GND_{RIO1PLL}$	**RapidIO Interface 1 PLL Ground** Ground dedicated for RapidIO Interface 1 PLL use. The connection should be provided with an extremely low-impedance path to ground.
	SR2_PLL_AGND	$GND_{RIO2PLL}$	**RapidIO Interface 2 PLL Ground** Ground dedicated for RapidIO interface 2 PLL use. The connection should be provided with an extremely low-impedance path to ground.
	SXCVSS1	GND_{SXC1}	**RapidIO Interface 1 Core Ground** A ground for the RapidIO port 1 Core circuitry.
	SXCVSS2	GND_{SXC2}	**RapidIO Interface 2 Core Ground** A ground for the RapidIO port 2 Core circuitry.
	SXPVSS1	GND_{SXP1}	**RapidIO Interface 1 Pad Ground** A ground for the RapidIO port 1 Pad circuitry.
	SXPVSS2	GND_{SXP2}	**RapidIO Interface 2 Pad Ground** A ground for the RapidIO port 2 Pad circuitry.

Note: The external decoupling capacitors recommendations are listed in the *MSC8156E Technical Data Sheet*.

3.2 Clock Signals

Table 3-4. Clock Signals

Signal Name	Type	Signal Description
CLKIN	Input	**Clock In** Primary clock input to the MSC8156E PLLs.
CLKOUT	Output	**Clock Out** The bus clock output.

3.3 Reset and Configuration Signals

Table 3-5. Reset and Configuration Signals

Signal Name	Type	Signal Description
$\overline{\text{PORESET}}$	Input	**Power-On Reset** When asserted, this line causes the MSC8156E to enter power-on reset state. Internally, this signal also resets the TAP and debugging modules. The power-on reset flow resets the MSC8156E device, configures various device attributes including its clock modes, and drives $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ as open-drain outputs.
$\overline{\text{HRESET}}$	Input/ Output	**Hard Reset** When asserted as an input, this signal causes the MSC8156E to abort all current internal and external transactions, set most registers to their default state, and enter the hard reset state. This signal must be asserted for at least 32 CLKIN cycles. While the device is in the hard reset state, it drives $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ as open-drain outputs. This signal requires an external pull-up resistor. The signal is tri-stated after the hard reset flow is complete.
$\overline{\text{SRESET}}$	Input/ Output	**Soft Reset** When asserted as an input, this signal causes the MSC8156E to enter the soft reset state, about all current internal transactions, configure most registers with their default values, and cause the cores to enter their reset state. The signal does not affect I/O signal functionality or direction or memory controller operations. While the device is in the soft reset state, it drives the $\overline{\text{SRESET}}$ as an open-drain output. This signal requires an external pull-up resistor. The signal is tri-stated after the soft reset flow is complete.
STOP_BS	Input	**Stop Boot Sequencer** This signal is valid only when the reset configuration words are being loaded from an I^2C EEPROM using the boot sequencer and is asserted only for a reset target device to prevent the loading of the reset configuration words until allowed by the Boot ROM. The signal level must be asserted as long as $\overline{\text{HRESET}}$ is asserted. For the reset master or a single device reading from I^2C EEPROM, you must drive this low during the reset sequence. For details, see **Chapter 5**, *Reset*. This signal is also used for booting after reset (for details, see **Chapter 6**, *Boot Program*).

Table 3-5. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
RCW_SRC0	Input	**Reset Configuration Word Source 0** Along with the RCW_SRC[1–2], this signal is sampled at the deassertion of PORESET to identify the source of the reset configuration word. The required signal level must be maintained as long as HRESET is asserted.
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
RCW_SRC1	Input	**Reset Configuration Word Source 1** Along with the RCW_SRC[0, 2], this signal is sampled at the deassertion of PORESET to identify the source of the reset configuration word. The required signal level must be maintained as long as HRESET is asserted.
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
RCW_SRC2	Input	**Reset Configuration Word Source 2** Along with the RCW_SRC[0–1], this signal is sampled at the deassertion of PORESET to identify the source of the reset configuration word. The required signal level must be maintained as long as HRESET is asserted.
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 21**, *Timers*.
RC[0–2]	Input	**Reset Configuration Word Bit 0–2** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO[0–2]	Input/ Output	**General-Purpose Input Output 0–2** Three of the 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.
IRQ[0–2]	Input	**Interrupt Request 0–2** External lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.

Table 3-5. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
RC3	Input	**Reset Configuration Word Bit 3** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO3	Input/ Output	**General-Purpose Input Output 3** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
IRQ3	Input	**Interrupt Request 3** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DRQ1	Input	**DMA External Request 1** When enabled by GPIO multiplexing, asserting this input triggers external DMA request 1. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC4	Input	**Reset Configuration Word Bit 4** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
IRQ4	Input	**Interrupt Request 4** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DDN1	Output	**DMA External DONE Indication 1** When enabled by GPIO multiplexing, this signal is asserted to indicate that DMA request 1 is done. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC[5–13]	Input	**Reset Configuration Word Bit 5–13** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO[5–13]	Input/ Output	**General-Purpose Input Output 5–13** Nine of the 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.
IRQ[5–13]	Input	**Interrupt Request 5–13** External lines that can request a service routine via the internal interrupt controller. Selected through GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 13**, *Interrupt Handling*.
RC14	Input	**Reset Configuration Word Bit 14** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
IRQ14	Input	**Interrupt Request 14** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DRQ0	Input	**DMA External Request 0** When enabled by GPIO multiplexing, asserting this input triggers external DMA request 0. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,

Table 3-5. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
RC15	Input	**Reset Configuration Word Bit 15** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
IRQ15	Input	**Interrupt Request 15** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DDN0	Output	**DMA External DONE Indication 0** When enabled by GPIO multiplexing, this signal is asserted to indicate that DMA request 0 is done. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC16	Input	**Reset Configuration Word Bit 16** Sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
GPIO16	Input/ Output	**General-Purpose Input Output 16** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.
RC17	Input	**Reset Configuration Word Bit 17** If RCW_SRC[0–2] equals 011, this input is sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
RCW_LSEL0	Output	**Reset Configuration Word Lane 0 Select** If RCW_SRC[0–2] equals 000, this signal is used to enable loading of Lane 0 (RCWLR bits 15–0) of the RCW via RC[15–0] when asserted. See **Chapter 5**, *Reset* for details.
RC18	Input	**Reset Configuration Word Bit 18** If RCW_SRC[0–2] equals 011, this input is sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
RCW_LSEL1	Output	**Reset Configuration Word Lane 1 Select** If RCW_SRC[0–2] equals 000, this signal is used to enable loading of Lane 1 (RCWLR bits 31–16) of the RCW via RC[15–0] when asserted. See **Chapter 5, *Reset*** for details.
RC19	Input	**Reset Configuration Word Bit 19** If RCW_SRC[0–2] equals 011, this input is sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
RCW_LSEL2	Output	**Reset Configuration Word Lane 2 Select** If RCW_SRC[0–2] equals 000, this signal is used to enable loading of Lane 2 (RCWHR bits 15–0) of the RCW via RC[15–0] when asserted. See **Chapter 5, *Reset*** for details.
RC20	Input	**Reset Configuration Word Bit 20** If RCW_SRC[0–2] equals 011, this input is sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers.
RCW_LSEL3	Output	**Reset Configuration Word Lane 3 Select** If RCW_SRC[0–2] equals 000, this signal is used to enable loading of Lane 3 (RCWHR bits 31–16) of the RCW via RC[15–0] when asserted. See **Chapter 5, *Reset*** for details.

Table 3-5. Reset and Configuration Signals (Continued)

Signal Name	Type	Signal Description
RC21	Input	**Reset Configuration Word Bit 21** If RCW_SRC[0–2] equals 011, this input is sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers. If RCW_SRC[0–2] does not equal 011, this input is ignored.
Note: When RCW_SRC[0–2] equals 011, RC[0–21] are valid only for driving a reduced external reset configuration word value. The signals are sampled during the assertion of PORESET to set part of the bits of the Reset Configuration Word Registers. The required signal levels must be maintained as long as HRESET is asserted. All other signal drivers connected to these inputs must be tri-stated while HRESET is asserted. When RCW_SRC[0–2] equals 000, the device loads all 64 bits of the RCW via RC[0–15] in four beats. In this case, RC[17–20] function as RCW_LSEL[0–3] outputs that are asserted to load the RCW 16 bits at a time and RC21 is ignored. See **Chapter 5**, *Reset* for details.		

3.4 Memory Controller 1 and 2

Refer to **Chapter 12**, *DDR SDRAM Memory Controller* for details on configuring these signals.

Table 3-6. Memory Controller Signals

Signal Name	Type	Description
M1A[15–0] M2A[15–0]	Output	**Address Bus** The memory interface address bus used to connect to external memory devices. MA0 is the lsb of the address driven by the DDR controller.
M1BA[2–0] M2BA[2–0]	Output	**Bank Address** Selects the DDR DRAM bank. Each DDR SDRAM can support four or eight logically addressable sub-banks. MBA0 must connect to bit zero of the SDRAM input bank address. This line is asserted during the mode register set command to specify the extended mode register.
M1DQ[63–0] M2DQ[63–0]	Input/ Output	**Data Bus** The MSC8156E device drives the bus during write cycles and the external memory drives the bus during read cycles.
M1DM[8–0] M2DM[8–0]	Output	**DDR SDRAM Data Output Mask** Masks unwanted data bytes transferred during a burst write. These signals are used to support sub-burst-size transactions (such as single-byte writes) on SDRAM in which all transactions occur in multi-byte bursts. MDM0 corresponds to the MSB and MDM7 corresponds to the LSB. MDM8 acts as the ECC data mask.
M1DQS[8–0] M2DQS[8–0]	Input/Output	**DDR SDRAM DQS** Strobe for byte-lane data capture. The signals are inputs driven by the DDR SRAM with read data and outputs driven by the DDR controller with write data. The data strobes may be single-ended or differential. Bit 8 is used as the ECC data mask.
M1DQS[8–0] M2DQS[8–0]	Input/Output	**DDR SDRAM DQS Complement** Complement strobe for byte-lane data capture. The signals are inputs driven by the DDR SRAM with read data and outputs driven by the DDR controller with write data. The data strobes may be single-ended or differential.
M1ECC[7–0] M2ECC[7–0]	Input/Output	**DDR Error Checking and Correcting Codes** As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes.

Table 3-6. Memory Controller Signals (Continued)

Signal Name	Type	Description
M1CK[2–0] M2CK[2–0]	Output	**DDR Clock Out** The DDR clock output. Each signal is part of a differential pair.
$\overline{\text{M1CK[2–0]}}$ $\overline{\text{M2CK[2–0]}}$	Output	**DDR Clock Out Inverted** The inverted DDR clock. Each signal is part of a differential pair.
M1CKE[1–0] M2CKE[1–0]	Output	**Clock Enable** When asserted, this signal enables the DDR clock for the DDR DRAM.
$\overline{\text{M1RAS}}$ $\overline{\text{M2RAS}}$	Output	**Row Address Strobe** Connects to DDR DRAM $\overline{\text{RAS}}$ input. This line is asserted for activate commands and is used for mode register set and refresh commands.
$\overline{\text{M1CAS}}$ $\overline{\text{M2CAS}}$	Output	**Column Address Strobe** Connects to DDR DRAM $\overline{\text{CAS}}$ input. This line is asserted for read or write transactions and for mode register set, refresh, and precharge commands.
$\overline{\text{M1WE}}$ $\overline{\text{M2WE}}$	Output	**Write Enable** Connects to DDR DRAM $\overline{\text{WE}}$ input.
$\overline{\text{M1CS[0–1]}}$ $\overline{\text{M2CS[0–1]}}$	Output	**Chip Select 0–1** Enables specific memory devices or peripherals connected to the bus.
M1MDIC[0–1] M2MDIC[0–1]	Input/Output	**Driver Impedance Calibration** These lines are used for automatic calibration of the DDR I/O.
M1ODT[0–1] M2ODT[0–1]	Output	**On-Die Termination** Memory controller outputs for the ODT to the SDRAM. Each signal represents the corresponding chip select.
M1APAR_OUT M2APAR_OUT	Output	**Parity Output** If enabled, drives the parity bit for the bus.
$\overline{\text{M1APAR\_IN}}$ $\overline{\text{M2APAR\_IN}}$	Input	**Parity Input** Receives the error indication from an open-drain parity error signal.

3.5 SerDes Multiplexed Signals for the Serial RapidIO, PCI Express, and SGMII Interfaces

Refer to **Chapter 5**, *Reset*, **Chapter 17**, *PCI Express Controller*, **Chapter 16**, *Serial RapidIO Controller*, and the *QUICC Engine Block Reference Manual with Protocol Interworking* for configuration information.

Table 3-7. SerDes Multiplexed Signals

Signal Name	Type	Description
SR1_IMP_CAL_RX	Input	**Serial RapidIO Controller 1 Receiver Impedance Control Signal** Receiver impedance calibration control signal.
SR1_IMP_CAL_TX	Input	**Serial RapidIO Controller 1 Transmitter Impedance Control Signal** Transmitter impedance calibration control signal.
SR1_RXD0	Input	**Serial RapidIO Controller 1 Receive Data 0** Serial data input for a x1 or x4 link. Each signal is part of a differential pair.

Table 3-7. SerDes Multiplexed Signals (Continued)

Signal Name	Type	Description
$\overline{\text{SR1\_RXD0}}$	Input	**Serial RapidIO Controller 1 Receive Data 0 Inverted** Inverted serial data input for a x1 or x4 link. Each signal is part of a differential pair.
SR1_RXD1	Input	**Serial RapidIO Controller 1 Receive Data 1** Serial data input for a x4 link. Each signal is part of a differential pair.
$\overline{\text{SR1\_RXD1}}$	Input	**Serial RapidIO Controller 1 Receive Data 1 Inverted** Inverted serial data input for a x4 link. Each signal is part of a differential pair.
SR1_RXD2	Input	**Serial RapidIO Controller 1 Receive Data 2** Serial data input for a x4 link. Each signal is part of a differential pair.
SG1_RX	Input	**Ethernet 1 SGMII Receive Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
$\overline{\text{SR1\_RXD2}}$	Input	**Serial RapidIO Controller 1 Receive Data 2 Inverted** Inverted serial data input for a x4 link. Each signal is part of a differential pair.
$\overline{\text{SG1\_RX}}$	Input	**Ethernet 1 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR1_RXD3	Input	**Serial RapidIO Controller 1 Receive Data 3** Serial data input for a x4 link. Each signal is part of a differential pair.
SG2_RX	Input	**Ethernet 2 SGMII Receive Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
$\overline{\text{SR1\_RXD3}}$	Input	**Serial RapidIO Controller 1 Receive Data 3 Inverted** Inverted serial data input for a x4 link. Each signal is part of a differential pair.
$\overline{\text{SG2\_RX}}$	Input	**Ethernet 2 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR1_TXD0	Output	**Serial RapidIO Controller 1 Transmit Data 0** Serial data output for a x1 or x4 link. Each signal is part of a differential pair.
$\overline{\text{SR1\_TXD0}}$	Output	**Serial RapidIO Controller 1 Transmit Data 0 Inverted** Inverted serial data output for a x1 or x4 link. Each signal is part of a differential pair.
SR1_TXD1	Output	**Serial RapidIO Controller 1 Transmit Data 1** Serial data output for a x4 link. Each signal is part of a differential pair.
$\overline{\text{SR1\_TXD1}}$	Output	**Serial RapidIO Controller 1 Transmit Data 1 Inverted** Inverted serial data output for a x4 link. Each signal is part of a differential pair.
SR1_TXD2	Output	**Serial RapidIO Controller 1 Transmit Data 2** Serial data output for a x4 link. Each signal is part of a differential pair.
SG1_TX	Output	**Ethernet 1 SGMII Transmit Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-7. SerDes Multiplexed Signals (Continued)

Signal Name	Type	Description
$\overline{\text{SR1\_TXD2}}$	Output	**Serial RapidIO Controller 1 Transmit Data 2 Inverted** Inverted serial data output for a x4 link. Each signal is part of a differential pair.
$\overline{\text{SG1\_TX}}$	Output	**Ethernet 1 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR1_TXD3	Output	**Serial RapidIO Controller 1 Transmit Data 3** Serial data output for a x4 link. Each signal is part of a differential pair.
SG2_TX	Output	**Ethernet 2 SGMII Transmit Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
$\overline{\text{SR1\_TXD3}}$	Output	**Serial RapidIO Controller 1 Transmit Data 3 Inverted** Inverted serial data output for a x4 link. Each signal is part of a differential pair.
$\overline{\text{SG2\_TX}}$	Output	**Ethernet 2 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR1_REF_CLK	Input	**Serial RapidIO Controller 1 Reference Clock** Reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
$\overline{\text{SR1\_REF\_CLK}}$	Input	**Serial RapidIO Controller 1 Reference Clock Inverted** Inverted reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
SR2_IMP_CAL_RX	Input	**Serial RapidIO Controller 2 Receiver Impedance Control Signal** Receiver impedance calibration control signal.
SR2_IMP_CAL_TX	Input	**Serial RapidIO Controller 2 Transmitter Impedance Control Signal** Transmitter impedance calibration control signal.
SR2_RXD0	Input	**Serial RapidIO Controller 2 Receive Data 0** Serial data input for a x1 or x4 link. Each signal is part of a differential pair.
PE_RXD0	Input	**PCI Express Receive Data 0** Serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SR2\_RXD0}}$	Input	**Serial RapidIO Controller 2 Receive Data 0 Inverted** Inverted serial data input for a x1 or x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_RXD0}}$	Input	**PCI Express Receive Data 0 Inverted** Inverted serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SR2_RXD1	Input	**Serial RapidIO Controller 2 Receive Data 1** Serial data input for a x4 link. Each signal is part of a differential pair.
PE_RXD1	Input	**PCI Express Receive Data 1** Serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SG2_RX	Input	**Ethernet 2 SGMII Receive Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-7. SerDes Multiplexed Signals (Continued)

Signal Name	Type	Description
$\overline{\text{SR2\_RXD1}}$	Input	**Serial RapidIO Controller 2 Receive Data 1 Inverted** Inverted serial data input for a x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_RXD1}}$	Input	**PCI Express Receive Data 1 Inverted** Inverted serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SG2\_RX}}$	Input	**Ethernet 2 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR2_RXD2	Input	**Serial RapidIO Controller 2 Receive Data 2** Serial data input for a x4 link. Each signal is part of a differential pair.
PE_RXD2	Input	**PCI Express Receive Data 2** Serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SG1_RX	Input	**Ethernet 1 SGMII Receive Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
$\overline{\text{SR2\_RXD2}}$	Input	**Serial RapidIO Controller 2 Receive Data 2 Inverted** Inverted serial data input for a x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_RXD2}}$	Input	**PCI Express Receive Data 2 Inverted** Inverted serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SG1\_RX}}$	Input	**Ethernet 1 SGMII Receive Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR2_RXD3	Input	**Serial RapidIO Controller 2 Receive Data 3** Serial data input for a x4 link. Each signal is part of a differential pair.
PE_RXD3	Input	**PCI Express Receive Data 3** Serial data input. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SR2\_RXD3}}$	Input	**Serial RapidIO Controller 2 Receive Data 3 Inverted** Inverted serial data input for a x4 link. Each signal is part of a differential pair.
$\overline{\text{PEI\_RXD0}}$	Input	**PCI Express Receive Data 1 Inverted** Inverted serial data input. Each signal is part of a differential pair. For details, see **Chapter 17, *PCI Express Controller***.
SR2_TXD0	Output	**Serial RapidIO Controller 2 Transmit Data 0** Serial data output for a x1 or x4 link. Each signal is part of a differential pair.
PE_TXD0	Output	**PCI Express Transmit Data 0** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.

Table 3-7. SerDes Multiplexed Signals (Continued)

Signal Name	Type	Description
$\overline{\text{SR2\_TXD0}}$	Output	**Serial RapidIO Controller 2 Transmit Data 0 Inverted** Inverted serial data output for a x1 or x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_TXD0}}$	Output	**PCI Express Transmit Data 0 Inverted** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SR2_TXD1	Output	**Serial RapidIO Controller 2 Transmit Data 1** Serial data output for a x1 or x4 link. Each signal is part of a differential pair.
PE_TXD1	Output	**PCI Express Transmit Data 1** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SG2_TX	Output	**Ethernet 2 SGMII Transmit Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
$\overline{\text{SR2\_TXD1}}$	Output	**Serial RapidIO Controller 2 Transmit Data 1 Inverted** Inverted serial data output for a x1 or x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_TXD1}}$	Output	**PCI Express Transmit Data 1 Inverted** Inverted serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SG2\_TX}}$	Output	**Ethernet 2 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
SR2_TXD2	Output	**Serial RapidIO Controller 2 Transmit Data 2** Serial data output for a x1 or x4 link. Each signal is part of a differential pair.
PE_TXD2	Output	**PCI Express Transmit Data 2** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SG1_TX	Output	**Ethernet 1 SGMII Transmit Data** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
$\overline{\text{SR2\_TXD2}}$	Output	**Serial RapidIO Controller 2 Transmit Data 2 Inverted** Inverted serial data output for a x1 or x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_TXD2}}$	Output	**PCI Express Transmit Data 2 Inverted** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SG1\_TX}}$	Output	**Ethernet 1 SGMII Transmit Data Inverted** Part of the Ethernet signals. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-7. SerDes Multiplexed Signals (Continued)

Signal Name	Type	Description
SR2_TXD3	Output	**Serial RapidIO Controller 2 Transmit Data 3** Serial data output for a x4 link. Each signal is part of a differential pair.
PE_TXD3	Output	**PCI Express Transmit Data 3** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
$\overline{\text{SR2\_TXD3}}$	Output	**Serial RapidIO Controller 2 Transmit Data 3 Inverted** Inverted serial data output for a x1 or x4 link. Each signal is part of a differential pair.
$\overline{\text{PE\_TXD3}}$	Output	**PCI Express Transmit Data 3 Inverted** Serial data output. Each signal is part of a differential pair. For details, see **Chapter 17**, *PCI Express Controller*.
SR2_REF_CLK	Input	**Serial RapidIO Controller 2 Reference Clock** Reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
$\overline{\text{SR2\_REF\_CLK}}$	Input	**Serial RapidIO Controller 2 Reference Clock Inverted** Inverted reference clock signal. Each signal is part of a differential pair. For SGMII operation, this provides the input clock.
Note: For proper definition of serial RapidIO modes (x1/x4), PCI Express, and SGMII, configure the interfaces using the Reset Configuration Word settings. For details, see **Chapter 5**, *Reset*.		

3.6 TDM and Ethernet Signals

The TDM and RGMII signals are listed together because they are multiplexed using the same signal lines. The GE1 and GE2 bits in the RCW control the selection between the TDM signals and the RGMII signals. See **Table 3-2** on page 3-2 for a detailed description of the multiplexing options. **Table 3-8** describes the signals in this group.

Table 3-8. TDM and Ethernet Signals

Signal Name	Type	Description
TDM3TDT	Input/ Output	**TDM3 Serial Transmitter Data** The serial transmit data signal for TDM 3. For configuration details, see **Chapter 19**, *TDM Interface*.
GE1_RD3	Input	**Ethernet 1 Receive Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM3TCK	Input	**TDM3 Transmit Clock** Transmit Clock for TDM 3. For configuration details, see **Chapter 19**, *TDM Interface*.
GE1_RD2	Input	**Ethernet 1 Receive Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM3TSN	Input/ Output	**TDM3 Transmit Frame Sync** Transmit frame sync for TDM 3. See **Chapter 19, *TDM Interface***.
GE1_RX_CLK	Input	**Ethernet 1 Receive Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-8. TDM and Ethernet Signals (Continued)

Signal Name	Type	Description
TDM3RDT	Input/ Output	**TDM3 Serial Receiver Data** The receive data signal for TDM 3. For configuration details, see **Chapter 19**, *TDM Interface*.
GE1_RD0	Input	**Ethernet 1 Receive Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM3RCK	Input/ Output	**TDM3 Receive Clock** The receive clock signal for TDM 3. For configuration details, see **Chapter 19**, *TDM Interface*.
GE1_GTX_CLK	Output	**Ethernet 1 Output Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM3RSN	Input/ Output	**TDM3 Receive Frame Sync** The receive sync signal for TDM 3. For configuration details, see **Chapter 19**, *TDM Interface*.
GE1_RD1	Input	**Ethernet 1 Receive Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM2TDT	Input/ Output	**TDM2 Serial Transmitter Data** The transmit data signal for TDM 2. For configuration details, see **Chapter 19, *TDM Interface***.
GE1_TX_CLK	Input	**Ethernet 1 Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM2TCK	Input	**TDM 2 Transmit Clock** Transmit clock for TDM 2. For configuration details, see **Chapter 19, *TDM Interface***.
GE1_TD3	Output	**Ethernet 1 Transmit Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM2TSN	Input/ Output	**TDM2 Transmit frame Sync** Transmit frame sync for TDM 2. For configuration details, see **Chapter 19**, *TDM Interface*.
GE1_TX_CTL	Output	**Ethernet 1 Transmit Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM2RDT	Input/ Output	**TDM2 Serial Receiver Data** The receive data signal for TDM 2. For configuration details, see **Chapter 19, *TDM Interface***.
GE1_TD1	Output	**Ethernet 1 Transmit Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM2RCK	Input/ Output	**TDM2 Receive Clock** The receive clock signal for TDM 2. For configuration details, see **Chapter 19, *TDM Interface***.
GE1_TD0	Output	**Ethernet 1 Transmit Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM2RSN	Input/ Output	**TDM2 Receive Frame Sync** The receive sync signal for TDM 2. For configuration details, see **Chapter 19, *TDM Interface***.
GE1_TD2	Output	**Ethernet 1 Transmit Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM1TDT	Input/ Output	**TDM1 Serial Transmitter Data** The transmit data signal for TDM 1. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_TD0	Output	**Ethernet 2 Transmit Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-8. TDM and Ethernet Signals (Continued)

Signal Name	Type	Description
TDM1TCK	Input	**TDM1 Transmit Clock** Transmit clock for TDM 1. For configuration details, see **Chapter 19**, *TDM Interface*.
GE2_RX_CLK	Input	**Ethernet 2 Receive Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM1TSN	Input/ Output	**TDM1 Transmit Frame Sync** Transmit frame sync for TDM 1. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_TD1	Output	**Ethernet 2 Transmit Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM1RDT	Input/ Output	**TDM1 Serial Receiver Data** The receive data signal for TDM 1. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_TX_CLK	Input	**Ethernet 2 Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM1RCK	Input/ Output	**TDM1 Receive Clock** The receive clock signal for TDM 1. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_RD1	Input	**Ethernet 2 Receive Data 1** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM1RSN	Input/ Output	**TDM1 Receive Frame Sync** The receive sync signal for TDM 1. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_RX_CTL	Input	**Ethernet 2 Receive Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM0TDT	Input/ Output	**TDM0 Serial Transmitter Data** The transmit data signal for TDM 0. For configuration details, see **Chapter 19**, *TDM Interface*.
GE2_TD3	Output	**Ethernet 2 Transmit Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM0TCK	Input	**TDM 0 Transmit Clock** Transmit Clock for TDM 0. For configuration details, see **Chapter 19**, *TDM Interface*.
GE2_GTX_CLK	Output	**Ethernet 2 Output Transmit Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM0TSN	Input/ Output	**TDM0 Transmit Frame Sync** Transmit Frame Sync for TDM 0. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_RD0	Input	**Ethernet 2 Receive Data 0** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM0RDT	Input/ Output	**TDM0 Serial Receiver Data** The receive data signal for TDM 0. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_RD3	Input	**Ethernet 2 Receive Data 3** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
TDM0RCK	Input/ Output	**TDM0 Receive Clock** The receive clock signal for TDM 0. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_RD2	Input	**Ethernet 2 Receive Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-8. TDM and Ethernet Signals (Continued)

Signal Name	Type	Description
TDM0RSN	Input/ Output	**TDM0 Receive Frame Sync** The receive sync signal for TDM 0. For configuration details, see **Chapter 19, *TDM Interface***.
GE2_TD2	Output	**Ethernet 2 Transmit Data 2** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE2_TX_CTL	Output	**Ethernet 2 Transmit Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE1_RX_CTL	Input	**Ethernet 1 Receive Control** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE_MDC	Output	**Ethernet Management Data Clock** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GE_MDIO	Input/ Output	**Ethernet Management Data Input/Output** For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

3.7 Serial Peripheral Interface (SPI) Signal Summary

Table 3-9 summarizes the Serial Peripheral Interface (SPI) signal lines, which are available in all modes.

Table 3-9. SPI Signals

Signal Name	Type	Signal Description
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. Valid in all modes.
$\overline{\text{SPI\_SL}}$	Input	**SPI Select** Enable input to the SPI slave in single master mode. In multi-master environment, $\overline{\text{SPI\_SL}}$ detects an error when more one master is operating. Assertion of an $\overline{\text{SPI\_SL}}$, while it is master, causes an error.
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. Valid in all modes
SPI_MISO	Input/ Output	**SPI Master Input Slave Output** When the SPI is a master, SPI_SCK is the clock input signal that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO.
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. Valid in all modes.
SPI_MOSI	Input/ Output	**SPI Master Output Slave Input** When the SPI is a master, SPI_SCK is the clock input signal that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO.
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. Valid in all modes.
SPI_SCK	Input/ Output	**SPI Clock** Gated clock, active only during data transfers. Four combinations of SPI_SCK phase and polarity can be configured. When the SPI is a master, SPI_SCK is the clock output signal that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO.

3.8 GPIO/Maskable Interrupt Signal Summary

Some of the GPIO and interrupt lines are multiplexed with other interfaces. Some of the lines are independent. In addition to the hardware interrupt inputs, there are also several signal lines used to reroute interrupts between the cores and an external host processor. **Table 3-16** summarizes the GPIO and interrupt signal lines.

Table 3-10. GPIO and Maskable Interrupt Summary

Signal Name	Type	Description
GPIO31	Input/ Output	**General-Purpose Input Output 31** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
I2C_SDA	Input/ Output	**I^2C-Bus Data Line** This is the data line for the I^2C bus. For details, see **Chapter 24**, *I2C*.

Table 3-10. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO30	Input/ Output	**General-Purpose Input Output 30** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
I2C_SCL	Input/ Output	**I^2C-Bus Clock Line** This the clock line for the I^2C bus. For details, see **Chapter 24**, *I2C*.
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
UART_TXD	Output	**UART Transmit Data** For details, see **Chapter 20**, *UART*.
GPIO28	Input/ Output	**General-Purpose Input Output 28** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
UART_RXD	Input/ Output	**UART Receive Data** For details, see **Chapter 20**, *UART*.
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
RCW_SRC0	Input	**Reset Configuration Word Source 0** Along with the RCW_SRC[1–2], this signal is sampled at the deassertion of $\overline{\text{PORESET}}$ to identify the source of the reset configuration word. The required signal level must be maintained as long as $\overline{\text{HRESET}}$ is asserted.
GPIO26	Input/ Output	**General-Purpose Input/Output 26** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. .
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
RCW_SRC1	Input	**Reset Configuration Word Source 1** Along with the RCW_SRC[0,2], this signal is sampled at the deassertion of $\overline{\text{PORESET}}$ to identify the source of the reset configuration word. The required signal level must be maintained as long as $\overline{\text{HRESET}}$ is asserted.

Table 3-10. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
RCW_SRC2	Input	**Reset Configuration Word Source 2** Along with the RCW_SRC[0–1], this signal is sampled at the deassertion of $\overline{\text{PORESET}}$ to identify the source of the reset configuration word. The required signal level must be maintained as long as $\overline{\text{HRESET}}$ is asserted.
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
TMR0	Input/ Output	**Timer 0** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO22	Input/ Output	**General-Purpose Input Output 22** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
GPIO21	Input/ Output	**General-Purpose Input Output 21** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
GPIO20	Input/ Output	**General-Purpose Input Output 20** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{SPI\_SL}}$	Input	**SPI Select** Enable input to the SPI slave in single master mode. In multi-master environment, $\overline{\text{SPI\_SL}}$ detects an error when more one master is operating. Assertion of an $\overline{\text{SPI\_SL}}$, while it is master, causes an error. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GPIO19	Input/ Output	**General-Purpose Input Output 19** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
SPI_MISO	Input/ Output	**SPI Master Input Slave Output** When the SPI is a master, SPI_SCK is the clock input signal that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GPIO18	Input/ Output	**General-Purpose Input Output 18** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
SPI_MOSI	Input/ Output	**SPI Master Output Slave Input** When the SPI is a master, SPI_SCK is the clock input signal that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.
GPIO17	Input/ Output	**General-Purpose Input Output 17** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
SPI_SCK	Input/ Output	**SPI Clock** Gated clock, active only during data transfers. Four combinations of SPI_SCK phase and polarity can be configured. When the SPI is a master, SPI_SCK is the clock output signal that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO. For details, see the *QUICC Engine Block Reference Manual with Protocol Interworking*.

Table 3-10. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO16	Input/ Output	**General-Purpose Input Output 16** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
RC16	Input	**Reset Configuration Word Bit 16** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ15}}$	Input	**Interrupt Request 15** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DDN0	Output	**DMA External DONE Indication 0** When enabled by GPIO multiplexing, this signal is asserted to indicate that DMA request 0 is done. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC15	Input	**Reset Configuration Word Bit 15** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ14}}$	Input	**Interrupt Request 14** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DRQ0	Input	**DMA External Request 0** When enabled by GPIO multiplexing, asserting this input triggers external DMA request 0. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC14	Input	**Reset Configuration Word Bit 14** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
GPIO[13–5]	Input/ Output	**General-Purpose Input Output 13–5** Nine of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ[13–5]}}$	Input	**Interrupt Request 13–5** External lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC[13–5]	Input	**Reset Configuration Word Bit 13–5** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.

Table 3-10. GPIO and Maskable Interrupt Summary (Continued)

Signal Name	Type	Description
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ4}}$	Input	**Interrupt Request 4** External lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DDN1	Output	**DMA External DONE Indication 1** When enabled by GPIO multiplexing, this signal is asserted to indicate that DMA request 1 is done. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC4	Input	**Reset Configuration Word Bit 4** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
GPIO3	Input/ Output	**General-Purpose Input Output 3** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ3}}$	Input	**Interrupt Request 3** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
DRQ1	Input	**DMA External Request 1** When enabled by GPIO multiplexing, asserting this input triggers external DMA request 1. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
RC3	Input	**Reset Configuration Word Bit 3** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
GPIO[2–0]	Input/ Output	**General-Purpose Input Output 2–0** Three of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ[2–0]}}$	Input	**Interrupt Request 2–0** External lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC[2–0]	Input	**Reset Configuration Word Bit 2–0** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.

3.9 Timer Signals

Table 3-11 describes the signals in this group.

Table 3-11. Timer Signals

Signal Name	Type	Description
TMR4	Input/ Output	**Timer 4** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO27	Input/ Output	**General-Purpose Input Output 27** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
RCW_SRC0	Input	**Reset Configuration Word Source 0** Along with the RCW_SRC[0–1], this signal is sampled at the deassertion of $\overline{\text{PORESET}}$ to identify the source of the reset configuration word. The required signal level must be maintained as long as $\overline{\text{HRESET}}$ is asserted.
TMR3	Input/ Output	**Timer 3** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO26	Input/ Output	**General-Purpose Input Output 26** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
TMR2	Input/ Output	**Timer 2** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO25	Input/ Output	**General-Purpose Input Output 25** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
RCW_SRC1	Input	**Reset Configuration Word Source 1** Along with the RCW_SRC[0,2], this signal is sampled at the deassertion of $\overline{\text{PORESET}}$ to identify the source of the reset configuration word. The required signal level must be maintained as long as $\overline{\text{HRESET}}$ is asserted.
TMR1	Input/ Output	**Timer 1** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO24	Input/ Output	**General-Purpose Input Output 24** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
RCW_SRC2	Input	**Reset Configuration Word Source 2** Along with the RCW_SRC[0–1], this signal is sampled at the deassertion of $\overline{\text{PORESET}}$ to identify the source of the reset configuration word. The required signal level must be maintained as long as $\overline{\text{HRESET}}$ is asserted.

Table 3-11. Timer Signals (Continued)

Signal Name	Type	Description
TMR0	Input/ Output	**Timer 0** Configured as input to the counter or output from the counter. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For timer functional details, see **Chapter 21**, *Timers*.
GPIO23	Input/ Output	**General-Purpose Input Output 23** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.

3.10 UART Signals

Table 3-12. UART Signals

Signal Name	Type	Description
UART_TXD	Output	**UART Transmit Data** Selected through the GPIO configuration. For details, see **Chapter 22, *GPIO***. For functional details, see **Chapter 20, *UART***.
GPIO29	Input/ Output	**General-Purpose Input Output 29** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
UART_RXD	Input/ Output	**UART Receive Data** Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 20**, *UART*.
GPIO28	Input/ Output\	**General-Purpose Input Output 28** One of 32 GPIO pins used as GPIO or as a dedicated input or output. For details, see **Chapter 22**, *GPIO*.

3.11 I^2C Signals

Table 3-13. I^2C Signals

Signal Name	Type	Description
I2C_SDA	Input/ Output	**I^2C-Bus Data Line** This is the data line for the I^2C bus. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 24**, *I2C*.
GPIO31	Input/ Output	**General-Purpose Input Output 31** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. Valid in all modes.
I2C_SCL	Input/ Output	**I^2C-Bus Clock Line** This the clock line for the I^2C bus. Selected through the GPIO configuration. For details, see **Chapter 22**, *GPIO*. For functional details, see **Chapter 24**, *I2C*.
GPIO30	Input/ Output	**General-Purpose Input/Output 30** One of 32 GPIOs. For details, see **Chapter 22**, *GPIO*. Valid in all modes.

3.12 External DMA Signals

Table 3-14. External DMA Signals

Signal Name	Type	Description
DDN0	Output	**DMA External DONE Indication 0** When enabled by GPIO multiplexing, this signal is asserted to indicate that DMA request 0 is done. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
GPIO15	Input/ Output	**General-Purpose Input Output 15** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ15}}$	Input	**Interrupt Request 15** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC15	Input	**Reset Configuration Word Bit 15** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
DRQ0	Input	**DMA External Request 0** When enabled by GPIO multiplexing, asserting this input triggers external DMA request 0. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
GPIO14	Input/ Output	**General-Purpose Input Output 14** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ14}}$	Input	**Interrupt Request 14** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC14	Input	**Reset Configuration Word Bit 14** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.
DDN1	Output	**DMA External DONE Indication 1** When enabled by GPIO multiplexing, this signal is asserted to indicate that DMA request 1 is done. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
GPIO4	Input/ Output	**General-Purpose Input Output 4** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ4}}$	Input	**Interrupt Request 4** External lines that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC4	Input	**Reset Configuration Word Bit 4** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.

Table 3-14. External DMA Signals (Continued)

Signal Name	Type	Description
DRQ1	Input	**DMA External Request 1** When enabled by GPIO multiplexing, asserting this input triggers external DMA request 1. For details, see **Chapter 14**, *Direct Memory Access (DMA) Controller*,
GPIO3	Input/ Output	**General-Purpose Input Output 3** One of the 32 GPIOs. For details, see **Chapter 22**, *GPIO*.
$\overline{\text{IRQ3}}$	Input	**Interrupt Request 3** External line that can request a service routine via the internal interrupt controller. For details, see **Chapter 13**, *Interrupt Handling*.
RC3	Input	**Reset Configuration Word Bit 3** Sampled during the assertion of $\overline{\text{PORESET}}$ to set part of the bits of the Reset Configuration Word Registers.

3.13 Other Interrupt Signals

Table 3-15 summarizes the other interrupt signal lines.

Table 3-15. Other Interrupt Signals

Signal Name	Type	Description
$\overline{\text{INT\_OUT}}$	Output	**Interrupt Output** An open-drain output driven from the MSC8156E virtual interrupt 24. Assertion of this output indicates that an unmasked interrupt is pending in the MSC8156E internal interrupt controller.
$\overline{\text{NMI}}$	Input	**Non-Maskable Interrupt** An external device may assert this line to generate a non-maskable interrupt to the MSC8156E device.
$\overline{\text{NMI\_OUT}}$	Output	**Non-Maskable Interrupt Output** An open-drain pin driven from the MSC8156E virtual interrupt 25. Assertion of this output indicates that a non-maskable interrupt is pending in the MSC8156E internal interrupt controller, waiting to be handled by an external host.

3.14 OCE Event and JTAG Test Access Port Signals

The MSC8156E uses two sets of debugging signals for the two types of internal debugging modules: OCE and the JTAG TAP controller. Each internal SC3850 core has an OCE module, but they are all accessed externally by the same two signals EE0 and EE1. The MSC8156E supports the standard set of test access port (TAP) signals defined by **IEEE**® Std. 1149.1™ Test Access Port and Boundary-Scan Architecture specification and described in **Table 3-16**.

Table 3-16. JTAG TAP Signals

Signal Name	Type	Signal Description
EE0	Input	**OCE Event Bit 0** Used for putting the internal SC3850 cores into Debug mode. Pulling the signal high asserts the signal and requests that the cores enter Debug mode.
EE1	Output	**OCE Event Bit 1** Indicates that at least one on-chip SC3850 core is in Debug mode. A high output indicates that at least one SC3850 core is in Debug mode.
TCK	Input	**Test Clock** A test clock signal for synchronizing JTAG test logic.
TDI	Input	**Test Data Input** A test data serial signal for test instructions and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TDO	Output	**Test Data Output** A test data serial signal for test instructions and data. TDO can be tri-stated. The signal is actively driven in the shift-IR and shift-DR controller states and changes on the falling edge of TCK.
TMS	Input	**Test Mode Select** Sequences the test controller's state machine, is sampled on the rising edge of TCK, and has an internal pull-up resistor.
$\overline{\text{TRST}}$	Input	**Test Reset** Asynchronous JTAG reset input. Initializes the TAP logic. This signal should always be asserted with $\overline{\text{PORESET}}$.

Chip-Level Arbitration and Switching System (CLASS) 4

The Chip Level Arbitration and Switching System (CLASS) is the central internal interconnect system for the MSC8156E device. The CLASS is a non-blocking, full-fabric interconnect that allows any initiator to access any target in parallel with another initiator-target couple. The CLASS uses a fully pipelined low latency design. The CLASS demonstrates per-target prioritized round-robin arbitration, highly optimized to the target characteristics. The CLASS operates at 500 MHz, and is separate from the SC3850 core frequency to provide an optimized trade-off between power dissipation, memory technology, and miss latency. Controlling the intradevice data flow, the CLASS reduces bottle necks and permits high bandwidth fully pipe-lined traffic. The CLASS system is ready for use and does not require any special configuration to perform non-blocking pipelined transactions from any initiator to any memory. The configurable arbitration features described in this chapter are for fine-tuning the system for specific application requirements.

The twelve CLASS initiators are:

- Six SC3850 core subsystems (initiator ports 0–5)
- Two MAPLE bridges (initiator ports 6–7)
- SerDes bridge shared by two Serial RapidIO controllers and the PCI Express controller (initiator port 8)
- Peripherals bridge shared by the SEC, four TDM interfaces, SPI, RGMII, SGMII, and JTAG interface (initiator port 9)
- Two DMA controllers (initiator ports 10–11)

The eight CLASS targets are:

- Configuration Control and Status Registers (CCSR) (target port 0)
- MAPLE module (target port 1)
- Three core subsystem bridges (two core subsystems per bridge) (target ports 2–4)
- Two DDR controllers (target ports 5–6)
- M3 memory (target port 7)

The CLASS initiators and targets are shown in **Figure 4-1**. The arrows indicate the address direction from initiator to target.

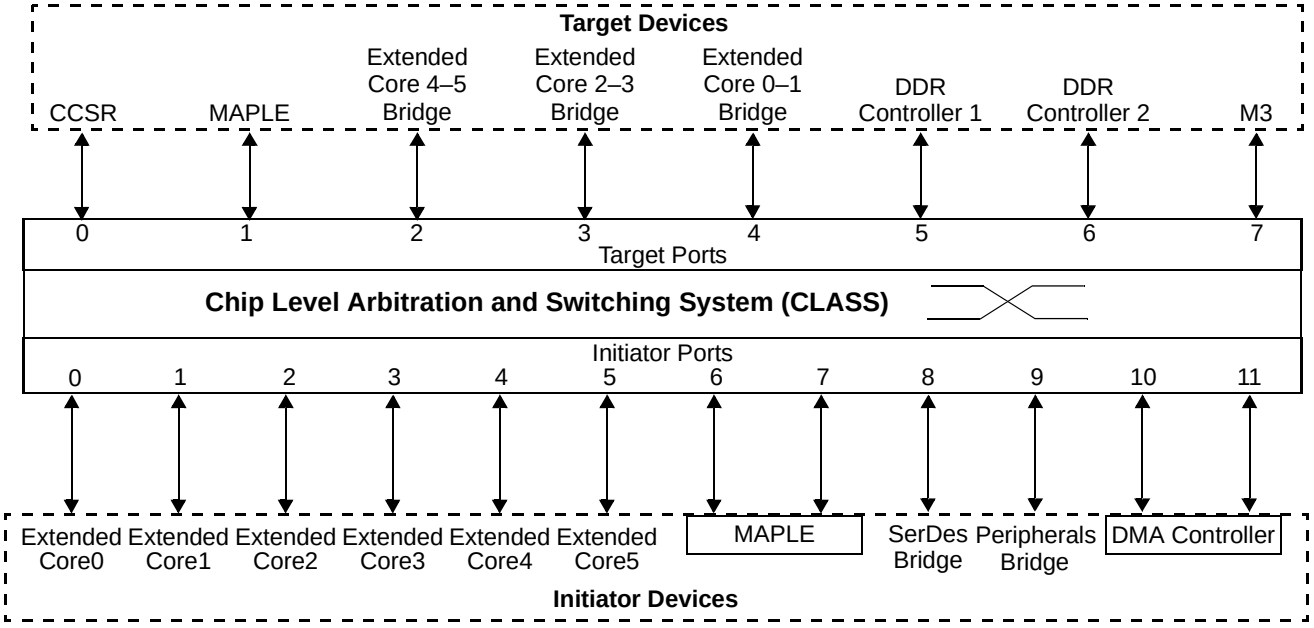


Figure 4-1. CLASS Initiators and Targets in the MSC8156E Device

4.1 CLASS Features

The CLASS modules implement the following features:

- Non blocking, full fabric interconnect.
- Full bandwidth utilization toward each of the targets.
- Allows full pipeline when a specific initiator accesses a specific target.
- Allows full pipeline when accesses are generated by one or more initiators to specific targets.
- Read transactions can have a maximum pipeline of 16 acknowledged requests before completing the transaction toward the initiator.
- Write transactions can have a maximum pipeline of 3 acknowledged requests before completing the transaction toward the initiator.
- Programmable priority mapping.
- Programmable auto priority upgrade.
- Address decoding for target selection and multi target demultiplexing:
  - Programmable address space start/end registers per target, for flexible address decoding (resolution of 4 KB). Not supported in the reduced configuration option.
  - Fixed priority between address decoding results which allows overlapping address windows and deduction of address windows.

- Per-target arbitration algorithm:
  - 4 level prioritization
  - Each level implements pseudo round-robin arbitration algorithm.
  - Weighted arbitration
  - Optimized data bus utilization mode
- Programmable masking priority for starvation elimination.
- Multiplexing the initiator buses according to the arbitration winner.
- Normalizing mode that splits non-aligned transactions according to the target capabilities (maximal burst size, power-of-2 burst, burst alignment, full size burst, data-beat alignment, wrap size)
- Error detection and handling:
  - The CLASS identifies illegal addresses; addresses that do not belong to any of the address windows or fall inside the negative windows.
  - The CLASS stores the illegal address, reports the error, and generates an interrupt.
- Debug and profiling unit (CDPU) support.

4.2 Functional Description

The CLASS is a non blocking interconnect between up to 16 initiators and up to 16 targets. The main sub-blocks of the CLASS are: expander, multiplexer and arbiter, normalizer, and the CLASS control interface (CCI) unit that implements the interface and interrupt lines and the CLASS register files. The CLASS also implements an inherent debug and profiling unit (CDPU).

To implement the protocol that deals with the point-to-point bus, the CLASS includes an expander module per initiator that performs address decoding and is used as sampling stage on the initiator side. Each expander module can detect an error address and generate an interrupt. For more details about the expander module see **Section 4.2.1**. From the target side, the CLASS includes a multiplexer and arbiter module and a normalizer module for each target. The multiplexer and arbiter module performs a pseudo round-robin (RR) arbitration algorithm between all the initiators and concentrates them toward one target. For details about multiplexer and arbiter module see **Section 4.2.2**. Each multiplexer and arbiter module has a dedicated normalizer module that is used as the sampling stage on the target side. The normalizer can also be used for normalizing transactions. For more details about normalizer module see **Section 4.2.3**.

4.2.1 Expander Module and Transaction Flow

Each expander module connects to one initiator. The expander module performs address decoding according to the configuration register settings. Each target is presented by a start address and an end address that define a window in the memory space. The address decoding is done by checking whether the transaction address hits one of the active windows. Each expander module is connected to all of the multiplexer and arbiter blocks in the CLASS to implement a full-fabric and non-blocking interconnect between any initiator to any target. If the address decoding hits in more than one window, the CLASS arbiter chooses a window by fixed priority arbitration (target 0 has the lowest priority). After detecting the requested target and the arbiter selects the target window, the expander module starts a transaction toward the associated multiplexer and arbiter module. The CLASS prevents the possibility of simultaneous accessing to more than one target by the same initiator. If there are accesses from one initiator to different targets, the expander module start the transactions to other targets only after all the open accesses to the current target are completed. The expander module is a sampling stage of transaction. For each request (address + attribute), write data is sampled from the initiator and driven to the normalizer module through multiplexer and arbiter module in the following clock cycle; read data is sampled from the normalizer module through multiplexer and arbiter module and driven to the initiator in the following clock cycle.

4.2.2 Multiplexer and Arbiter Module

The multiplexer and arbiter module connects to all the expander modules on one side and to a dedicated normalizer module on the other side. The multiplexer and arbiter module block is a pure logic data path design, that supports up to 16 initiators, performs an arbitration, and concentrates them towards a specific target normalizer module.

4.2.2.1 CLASS Arbiter

The CLASS arbiter performs weighted arbitration algorithm for requestors simultaneously using a pseudo round-robin arbitration algorithm for each of the priority levels and chooses the highest level request. The CLASS arbiter supports four priority levels, where 3 is the highest and 0 is the lowest. The arbitration operation can be done every clock cycle or delayed according to the number of datums of acknowledged transaction (Late Arbitration mode). The CLASS arbiter supports priority upgrade, so the initiator can upgrade the priority level at any clock cycle.

To eliminate starvation for initiators with low priority, the Masking Priority should be enabled. Starvation can occur when the higher priority initiators access continuously and the lower priority initiators can not perform any access (no priority upgrade ability by the initiator and auto priority upgrade in the expander module is disabled). When the Masking Priority is enabled, the arbiter dedicates slots for lower priority initiator in which the higher priority initiators are masked.

4.2.2.1.1 Weighted Arbitration

The CLASS arbiter supports limited weighted arbitration. Weighted arbitration is needed to apply non-uniform distribution of the bandwidth from all initiators toward each target. Weighted arbitration is configurable per CLASS target and gives configurable weights to each initiator. The CLASS arbiter ensures that when a weighted initiator wins the arbitration, it performs Weight + 1 consecutive transactions before transferring control to another initiator with the same or lower priority level.

4.2.2.1.2 Late Arbitration

In late arbitration mode, the request is initiated by the class arbiters as late as possible. At the end of a data burst, this can give better or worse performance for the initiators. The performance depends on the bursty character of the application and the utilization to the target. This mode is activated/deactivated by the appropriate bit in the C0ACR (see **Section 4.8.26**, *CLASS Arbitration Control Register (C0ACR)*).

4.2.2.1.3 Priority Masking

When C0ACR[PME] is set, the class arbiters are configured to preserve cycle slots for low priority accesses. They reserve 1/16 of all cycles for priority 0, 2/16 of all cycles for priority 1 or 0, and 2/16 of all cycles for priority 2, 1, or 0. This mode can decrease overall performance. This is one of two approaches to eliminate starvation. The other is to use auto-priority upgrade.

4.2.2.1.4 Auto Priority Upgrade

This mode is activated by setting the C0PACRx[AUE] bit (see **Section 4.8.3**, *CLASS Priority Auto Upgrade Control Registers (C0PACRx)*). When active, a pending request has its priority upgraded to the next higher priority after a specified number of cycles specified by C0PAVRx[AUV] (see **Section 4.8.2**, *CLASS Priority Auto Upgrade Value Registers (C0PAVRx)*). The upgrade level and timing depend on the current priority value assigned, as follows:

- For priority 0 requests, the priority is upgraded to priority 1 after AUV cycles.
- For priority 1 requests, the priority is upgraded to priority 2 after AUV/2 cycles.
- For priority 2 requests, the priority is upgraded to priority 3 (highest) after AUV/4 cycles.

The upgrade process continues until the request is processed or it reaches priority 3.

4.2.2.2 CLASS Multiplexer

The CLASS multiplexer includes two FIFOs that connect between the appropriate initiator and the target. The FIFO depth is 16, thus enabling the multiplexer and arbiter module to deal with 16 open transactions, which received their request acknowledge and are waiting for the end-of-data or end-of-transaction signals. The CLASS multiplexer is pure logic for the data path and does not cause any latency.

4.2.3 Normalizer Module

Each normalizer module is connected to the appropriate multiplexer and arbiter module on one side and to a specific CLASS target interface on the other side. Each normalizer module is used as a sampler for full pipeline towards the target. The normalizer module is the only module within the CLASS that can manipulate the transaction (for example, splitting non-aligned transactions). An internal signal is used to indicate that optimization is needed. Only the last normalizer module on the way to the target is used for normalization. All the other normalizer modules should be used only as samplers. The normalizer module supports the fast confirm mechanism for writes.

4.2.4 CLASS Control Interface (CCI)

The CLASS control interface (CCI) enables access to the CLASS configuration, control, and status registers. All write accesses to these registers should use supervisor mode. See **Section 4.8** for programming details.

4.3 MSC8156E Initiator CLASS Access Priorities

Table 4-1 summarizes the priorities of the CLASS initiators when accessing targets over the CLASS. The priority listed in this table is the priority of the accesses by the initiator before any CLASS remapping (that is, C0PMRx = 0x3210). If the CLASS is remapped (using C0PMRx), this table represents the arrival priority of the initiator to the CLASS and not the new mapped value. Note that start-up code (for example, the CodeWarrior initialization files) and boot code can change the value of C0PMRx from its reset value.

Table 4-1. Initiator CLASS Access Priorities

Initiator	Priority 0	Priority 1	Priority 2	Priority 3	comments
Cores and subsystems - Prefetch	“Low” priority (0)	NA	NA	NA	
Cores and subsystems - Read	NA	NA	“High” priority (2)	NA	A core read transaction reaches the CLASS in case of L1/L2 miss or non-cacheable address
Cores and subsystems - Write	“Low” priority (0) - Normal write	NA	“High” priority (2) - If the write stalls the core	NA	
MAPLE	(0)	NA	NA	NA	
SRIO - Inbound (IO, Maintenance, Doorbell)	SRIO priority (0)	SRIO priority (1)	SRIO priority (2)	NA	
SRIO - Inbound (Message)	NA	SRIO priority (0), SRIO priority (1)	SRIO priority (2)	NA	SRIO priority (0) and (1) both result in CLASS priority 1

Table 4-1. Initiator CLASS Access Priorities (Continued)

Initiator	Priority 0	Priority 1	Priority 2	Priority 3	comments
RMU - BD Read	NA	NA	Priority (2)	NA	
RMU - DATA Read (Outbound Doorbell or Message)	Priority (0), Priority (1)	Priority (2)	NA	NA	Serial RapidIO priority (0) and (1) both result in CLASS priority 0
OCNDMA BD Read	(0)	NA	NA	NA	
OCNDMA DATA Read/Write	(0)	NA	NA	NA	
System DMA BD Read	NA	NA	NA	(3)	
System DMA DATA Read/Write	PP (0)	PP (1)	PP (2)	PP (3)	See **Section 14.7.22.1**, *Buffer Attributes (BD_ATTR)* and **Section 14.7.22.2**, *Multi-Dimensional Buffer Attributes (BD_MD_ATTR)*.
PCI Express Inbound	NA	NA	(2)	NA	
TDM	NA	Default Priority (1)	NA	Emergency (Defined by progrmable threshold of TDM FIFO fill)	
QUICC Engine subsystem	Normal mode (0)	Normal mode (1)	Normal mode (2)	Normal/emergency mode (3)	See **Section 18.3.4**, *MBus Access* and the Serial DMA Mode Register (SDMR) in the *QUICC Engine Block Reference Manual with Interworking Protocol*.
SEC	(0)	(1)	(2)	(3)	See **Section 27.7.4.1**, MCR[PRIORITY] field

4.4 CLASS Error Interrupts

The CLASS can generate one error interrupt that is common for all initiators. The error interrupt is created when the CLASS receives a transaction request with an illegal address. Illegal addresses are defined as any one of the following 2 cases:

1. An address which does not belong to any of the address space windows of the enabled address decoders.
2. An address which falls within any of the address space windows of the enabled error address decoders.

When an illegal address is identified by the CLASS, the following events occur.

- The associated **AEIx** bit in the CISR is set.
- The address that was identified as illegal is stored in the associated **CEARx** and **CEEARx**. These registers are locked until the associated **AEIx** bit in the CISR is cleared either by a hardware reset or by writing 1 to this bit.

Note: If the associated **AEIx** bit in the CISR is already set when the illegal address is identified (due to a prior illegal address), then the new error address is not stored.

- If the corresponding **AEIEx** bit in the CIER is set, an IRQ is issued.
- The CLASS does not initiate a transaction to any target. However, the CLASS will continue normally on the initiator side until completion, and report the error. In case of a read transaction, the CLASS delivers invalid data to the initiator.
- If, at the time of the error transaction, there are open transactions that did not receive the end-of-transaction, the expander module stalls all new transaction until all prior transactions receive the end-of-transaction, close the error transaction, report the end-of-transaction, report the error, and only then continue with subsequent transactions.
- Any subsequent requests with a legal address are serviced normally.

Note: The CLASS does not produce an error when a transaction starts inside a target address window and finishes outside of the window. This situation must be avoided by the user. If it occurs, the results are unpredictable.

The error interrupt is logically ORed with internal error interrupts. The internal error interrupts are associated with each initiator. Thus, the CLASS error interrupt is asserted when at least one internal interrupt is asserted.

4.5 CLASS Debug Profiling Unit

The CLASS supports debug and profiling measurements by the CLASS debug and profiling unit (CDPU).

4.5.1 Profiling

The user can configure the desired measurement in the CIPCRs and CTPCRs.

Note: For each CLASS module, only one PMM field among all C0IPCRx and C0TPCRx can be greater than 0 during profiling.

You can activate the CDPU by:

- Writing a 1 to the CPCR[PE] bit.
- Configuring a watch point event in CPCR[WPEC] field.

The CDPU is deactivated by:

- Writing a 0 to the CPCR[PE] bit.
- Configuring a watch point event in CPCR[WPEC] field.
- Reaching a time-out in the CPTOR when the CPCR[TOE] bit is set.
- CPRCR overflow.

After the desired profiling mode has been chosen, activate the CDPU to perform the measurement. At the beginning of every measurement, the CLASS Profiling Reference Counter (CPRCR) starts counting the clock cycles. Read the CPISR[OVE] bit to verify that the measurement is complete and that the profiling counter values are valid. If the CPISR[OVE] is clear, read the profiling counters CPRCR and CPGCR and analyze the results.

4.5.2 Watch Point Unit

The CLASS includes a watch point unit (WPU) for each of the initiator interfaces and for each of the target interfaces. The WPU can compare programmed values to the real transactions and generate a watch point event when a match occurs.

Note: For each CLASS module, only one WPEN field can be set among all C0IWPCRx and C0TWPCRx when snooping watch point events. That is, only one watch point unit can be active at a time.

Use the following steps to use the watch point unit:

1. Clear C0PCR.
2. Clear C0IPCRx and C0TPCR.
3. For the time-out mechanism, program C0PTOR and set the C0PCR[TOE] bit.
4. Define the transaction to be monitored by writing the desired configuration to C0WPCR, C0WPACR, C0WPEACR, and C0WPAMR.
5. Enable the watch point units through C0IWPCRx and C0TWPCR.
6. Set the C0WPRCR[CE] bit to enable counting of the watch point events. If you use the watch point events to enable/disable the profiling unit according to WPCE, clear this bit.
7. After the measurement are finished check the following registers:
   - Read the C0PISR[OVE] bit.
   - In time-out mode, read C0PRCR.
   - If C0PISR[OVE] is set or if C0PRCR is equal to C0PTOR, the results are not valid.
   - Read C0PGCRx to get the number of watch point events during the measurement.

4.5.3 Event Selection

Events are selected using a combination of the CLASS watch point and profiling registers. **Table 4-2** lists the measurement modes, the required configuration settings, and the events measured by the specific CLASS Profiling General Registers for each CLASS module. See **Section 4.8** for the register details.

Table 4-2. C0PGCRx Events Selection

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	C0WPCR [CE]	C0TPCR [TT]	C0TPCR [PMM]	C0IPCRx [PMM]	C0PGCR0	C0PGCR1	C0PGCR2	C0PGCR3
None selected	0	—	00	00000	—	—	—	—
Initiator Priority and Auto-Upgrade	0	—	00	00001	No. of Initiator Requests with Priority 1	No. of Initiator Requests with Priority 2	No. of Initiator Requests with Priority 3	No. of Initiator Auto-Upgrade
	Allows the user to profile a statistical distribution of transaction priorities. This information can be used to consider whether other arbitration methods (priority mapping, Auto-upgrade, weighted arbitration, priority mask enable) should be considered							
Initiator Access Type	0	—	00	00010	Initiator Pending Request cycles (not acknow-ledged)	No. of Initiator Read Requests	No. of Initiator Real Write Requests (not confirmed)	No. of Initiator Fast Write Requests (not confirmed)
	Real/Fast confirmation refers to the type of eot for writes that are requested per MBus transaction. • Real means that for coherency reasons the eot for write should be high only after the data is written to the target. • Fast means that for performance reasons the eot can be high even before the data is written to the target. Real/Fast confirmation support is initiator dependent and the user cannot change the related settings. A summary follows below: • DSP Cores – Write through uses fast confirmation – Write through with SYNCIO generate real confirmation – Last burst in a line write-back is sent with real confirmation • DMA. Channel transfers come with fast confirm and real confirm on the last data of a transfer. • Serial RapidIO Inbound Transactions. When the transfer comes with a response (NWRITE_R), the last data uses real confirmation and fast confirm for the previous transfer. For NWRITE, data is transferred with fast confirmation • Serial RapidIO Outbound Transactions. Supports fast confirm on the last data transfer per channel transfer and fast confirm for the previous transfer. • QUICC Engine and PCI Express Transactions. Always uses real confirmation The resulting information can be used to redesign the code to minimize stalls related to real confirmations. Use of large transactions reduces the number of real confirmations because they are only required for the last beat of the transfer.							
Initiator Stall	0	—	00	00011	No. of Write After Read (WAR) events	No. of stall cycles due to WAR events	—	No. of stall cycles due to TS events
	By managing WAR and target switching, an initiator can enhance the performance for memory accesses and thus minimize run time. **Note:** Stall at the initiator does not mean that the initiator target data phase is idle; it indicates the delay after which the next access from the initiator starts at the target after a WAR or TS event.							

Table 4-2. C0PGCRx Events Selection (Continued)

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	C0WPCR [CE]	C0TPCR [TT]	C0TPCR [PMM]	C0IPCRx [PMM]	C0PGCR0	C0PGCR1	C0PGCR2	C0PGCR3
Initiator Priority Upgrade	0	—	00	00100	Initiator Sample 0 Upgrade cycles	Initiator Sample 1 Upgrade cycles	Initiator Sample 2 Upgrade cycles	—
	Initiator Priority Upgrade measurement counts the number of cycles a low-priority transaction is upgraded because a high-priority transaction was scheduled into the CLASS pipeline. This upgrading mechanism is necessary to reduce the service latency of newly issued transactions because the CLASS is ordered in nature and does not do preemption. A high-priority transaction could otherwise be delayed by preceding lower priority accesses for long periods of time. The upgrade is to the priority level of the high-priority transaction and it is only possible if the transaction is upgradable. The upgrade attribute of a transaction is initiator dependent and it is hard-wired. The system DMA is the only initiator issuing non-upgradable transactions. The counter measurements take place in the early stages of the CLASS pipeline at different sample locations. Transactions in sample 0 are older than transactions on sample 1, and transactions in sample 1 are older than the ones in sample 2. Any type of priority upgrade, including auto-upgrade, is captured by these counters. These counters provide a good view of the starvation dynamics of the system.							
Initiator Priority Non-Upgrade	0	—	00	00101	Initiator Sample 0 No Upgrade cycles	Initiator Sample 1 No Upgrade cycles	Initiator Sample 2 No Upgrade cycles	—
	Initiator Priority No-Upgrade measurement counts the number of cycles an older low-priority transaction is not upgraded despite the fact that a newer high-priority is scheduled by the same initiator. Compare with “Initiator Priority Upgrade”. This applies only to System DMA transactions because they are not upgradable by default. All other initiator transactions can be upgraded. These counts are not very useful for performance analysis, but they provide a good view of the starvation dynamics of the system.							
Initiator Supervisor	0	—	00	00110	Request pending cycles	No. of supervisor accesses	No. of non-supervisor accesses	—
	Supervisor/user accesses can be used to profile the amount of time the OS spends running and how much time is left for the users application. Effective for evaluating the core subsystem supervisor/user mode usage. The implementation depends on the values configured in M_DSDAx[DAPU]/M_DSDAx[DAPS], and so forth. See the *SC3850 Core Subsystem Reference Manual* for configuration details.							
Initiator Bandwidth	0	—	00	00111	No. of Initiator Read Data Acknow-ledges.	No. of Initiator Write Data Acknow-ledges	—	—
	Can be used to profile the number of data reads and writes. The amount of data that passes through the initiator port = [(NumberOfReadAck + NumberOfWriteAck) × W]. where W is the port width. Note that an access may be smaller than the port width.							
Initiator-target Bandwidth	0	—	00	10000 + T	No. of Read Data Acknow-ledges between Initiator and Target T	No. of Write Data Acknow-ledges between Initiator and Target T	—	—
	Can be used to profile the number of data reads and writes. between an initiator and a specific target The amount of data that passes through the initiator port = [(NumberOfReadAck + NumberOfWriteAck) × W]. where W is the port width. Note that an access may be smaller than the port width.							

Table 4-2. C0PGCRx Events Selection (Continued)

Measurement Mode	Configuration Settings for Each Mode				Events Measured			
	C0WPCR [CE]	C0TPCR [TT]	C0TPCR [PMM]	C0IPCRx [PMM]	C0PGCR0	C0PGCR1	C0PGCR2	C0PGCR3
Arbitration Winner Priority	0	0	01	00000	No. of priority 0 trans-actions at Target T	No. of priority 1 trans-actions at Target T	No. of priority 2 trans-actions at Target T	No. of priority 3 trans-actions at Target T
	Can be used to see the priority distribution for a target port							
Target Access Splitting	0	1	01	00000	No. of M-byte accesses toward target T. M = Initiator requests (pre splitting accesses)	No. of N-byte accesses toward target T. N = Initiator requests (post splitting accesses)	—	—
	Target access splitting gives statistical information about the ratio between initiator and target accesses towards every target. It returns the number of accesses in pre and post splitting. For example, say there are only 2 types of accesses to some target #T: 64-Byte and 16-Byte, and this target only supports 16-Byte accesses. Then, if the count shows 10000 initiator accesses and 34000 target accesses, this translates to 8000 × (64-Byte accesses) + 2000 × (16-Byte accesses), and they were split into 8000 × (4 × 16-Byte accesses) + 2000 × (16-Byte access) = 32000 + 2000 = 34000 accesses							
Arbitration Collision	0	0	10	00000	Target T Pending Request cycles	—	—	—
	This represents the number of cycles with more than one request directed to Target T.							
Target Bandwidth	0	1	10	00000	No. of Target T Read Data Acknow-ledges	No. of Target T Write Data Acknow-ledges	—	—
	Can be used to profile the number of data reads and writes to Target T. The amount of data that passes through the target port = [(NumberOfReadAck + NumberOfWriteAck) × W] where W is the port width.Note that an access can be less than the port width							
Target Stall	0	—	11	00000	No. of Write after Read (WAR) events	No. of stall cycles due to WAR events	—	—
	By managing WAR event, the system designer can enhance memory access performance and thus minimize run time. **Note:** A WAR stall at the target does not mean that the target bus is idle. It indicates the delay after which the next access from the initiator starts at the target after a WAR event.							
Watch Point	1	—	00	00000	No. of Watch Point Event	—	—	—
	Watch point event scan be snooped on any initiator and any target. It can be used for debug and also for triggering profiling counts that are pre-configured, this is non-intrusive (eliminating the need to write to the registers in the middle of an application)							

4.5.4 Debug and Profiling Events

The CLASS generates two event interrupts:

- Watch point event (WPE)
- Overflow event (OVE)

4.6 CLASS Reset

The CLASS implements 2 kinds of reset:

- Synchronous hard reset.
- Synchronous soft reset.

4.6.1 Soft Reset

This kind of reset has the following effects:

- All the CLASS state machines return to their idle state.
- All the CLASS operation FFs return to their idle state.
- The CLASS configuration registers are reset as described in the table for each register in **Section 4.8**, *Programming Model*.

4.6.2 Hard Reset

This reset brings all states machines to idle state and sets all CLASS registers to the reset values.

4.7 Limitations

- The CLASS does not support split transaction between targets. A split transaction starts inside a targets address space but ends outside of this window. The CLASS does not report an error in this event and the results are unpredictable. You must avoid this situation.
- The CLASS does not support pipelined transactions between different targets by the same initiator. The pipeline is stalled until all transaction to one target are closed before issuing a transaction to a different target.
- Arbitration Fairness. Requests with the higher priority levels may cause transactions with lower priority levels not to be acknowledged, resulting in a starvation condition. This situation can be prevented by using the auto priority upgrade supported by the expander module and/or by the multiplexer and arbiter module priority mask mechanism.
- Do not allow the TDM or QUICC Engine interfaces (Ethernet and SPI) to access the MAPLE-B block.
- Do not allow MAPLE-B to access itself.
- Do not allow cores to access each other.

4.8 Programming Model

All the CLASS registers are 32-bit registers. All the read and write accesses are executed through the bus. The CLASS modules use the following registers:

- CLASS Priority Mapping Registers (see page 4-16)
- CLASS Priority Auto Upgrade Value Registers (see page 4-18)
- CLASS Priority Auto Upgrade Control Registers (see page 4-19)
- CLASS Error Address Registers (see page 4-20)
- CLASS Error Extended Address Registers (see page 4-21)
- CLASS Initiator Profiling Configuration Registers (see page 4-23)
- CLASS Initiator Watch Point Control Registers (see page 4-25)
- CLASS Arbitration Weight Registers (see page 4-26)
- CLASS Start Address Decoder x (see page 4-27)
- CLASS End Address Decoder x (see page 4-28)
- CLASS Attributes Decoder x (see page 4-29)
- CLASS IRQ Status Register (see page 4-30)
- CLASS IRQ Enable Register (see page 4-31)
- CLASS Target Profiling Configuration Register (see page 4-32)
- CLASS Profiling Control Register (see page 4-33)
- CLASS Watch Point Control Register (see page 4-34)
- CLASS Watch Point Access Configuration Register (page 4-36)
- CLASS Watch Point Extended Access Configuration Register (see page 4-37)
- CLASS Watch Point Address Mask Register (see page 4-38)
- CLASS Profiling Time Out Register (see page 4-39)
- CLASS Target Watch Point Control Register (see page 4-40)
- CLASS Profiling IRQ Status Register (see page 4-41)
- CLASS Profiling IRQ Enable Register (see page 4-42)
- CLASS Profiling Reference Counter Register (see page 4-42)
- CLASS Profiling General Counter Registers (see page 4-43)
- CLASS Arbitration Control Register (see page 4-44)

Note: The base address for addressing CLASS registers is 0xFFF18000.

4.8.1 CLASS Priority Mapping Registers (C0PMRx)

C0PMR[0–11] CLASS Priority Mapping Registers Offset 0x800 + x*0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															PB
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	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
	—		PM3		—		PM2		—		PM1		—		PM0	
Type	R/W															
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0
Boot	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	1

C0PMRx is used as a look-up table for mapping the priority received from the initiator. By default the input priority is mapped to an identical value on the output. This register also enables/disables the priority derivation feature.

Note: You cannot write to this register while there are open CLASS transactions. The boot overrrides the default reset value for C0PMR[0–5] only.

Table 4-3 lists the C0PMRx bit field descriptions.

Table 4-3. C0PMRx Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to 0 for future compatibility.	
PB 16	0	**Priority Bypass** Enables/disables the priority derivation mechanism.	0 Filter the mapped priority with the priority derivation mechanism. 1 Pass the mapped priority from initiator to target with no filtering.
— 15–14	0	Reserved. Write to 0 for future compatibility.	
PM3 13–12	11	**Priority Mapping 3** Holds the priority value assigned to transactions that arrive with a value of 3.	00 Priority 0 01 Priority 1 10 Priority 2 11 Priority 3
— 11–10	0	Reserved. Write to 0 for future compatibility.	
PM2 9–8	10	**Priority Mapping 2** Holds the priority value assigned to transactions that arrive with a value of 2.	00 Priority 0 01 Priority 1 10 Priority 2 11 Priority 3
— 7–6	0	Reserved. Write to 0 for future compatibility.	
PM1 5–4	01	**Priority Mapping 1** Holds the priority value assigned to transactions that arrive with a value of 1.	00 Priority 0 01 Priority 1 10 Priority 2 11 Priority 3

Table 4-3. C0PMRx Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 3–2	0	Reserved. Write to 0 for future compatibility.	
PM0 1–0	0	**Priority Mapping 0** Holds the priority value assigned to transactions that arrive with a value of 0.	00 Priority 0 01 Priority 1 10 Priority 2 11 Priority 3

4.8.2 CLASS Priority Auto Upgrade Value Registers (C0PAVRx)

C0PAVR[0–11] CLASS Priority Auto Upgrade Value Registers Offset 0x840 + x*0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	AUV															
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C0PAVRx holds the value loaded to the priority auto-upgrade counter.

Note: You can write to this register while there are open CLASS transactions. The AUV field is loaded into the auto-upgrade counter only when you set the AUE bit in C0PACRx. Therefore, always update the AUV field in the C0PAVRx before you set the AUE bit.

Table 4-4 lists the C0PAVRx bit field descriptions.

Table 4-4. C0PAVRx Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to 0 for future compatibility.
AUV 15—0	0xFFFF	**Auto-Upgrade Value** The value loaded into the auto-upgrade counter. The priority of the access determines which bits of this value are used, as follows: • Priority 0: All 16 bits are loaded into the counter. • Priority 1: Bits 15–1 are loaded into bit 14–0 of the counter and a 0 into bit 15. • Priority 2: Bits 15–2 are loaded into bits 13–0 of the counter and 0 into bits 15 and 14.

4.8.3 CLASS Priority Auto Upgrade Control Registers (C0PACRx)

C0PACR[0–11] CLASS Priority Auto Upgrade Control Registers Offset 0x880 + x*0x04

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

C0PACRx controls the priority auto-upgrade mechanism.

Note: You can write to this register while there are open CLASS transactions.

Table 4-5 lists the C0PACRx bit field descriptions.

Table 4-5. C0PACRx Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to 0 for future compatibility.	
AUE 0	0	**Auto-Upgrade Enable** Enables/disables the auto-upgrade mechanism. **Note:** This bit can only be cleared by a hardware reset.	0 Auto-upgrade mechanism disabled. 1 Auto-upgrade mechanism enabled.

4.8.4 CLASS Error Address Registers (C0EARx)

C0EAR[0–11] CLASS Error Address Registers Offset 0x980 + x*0x04

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

The C0EAR is used to store the address (32 least significant bits) of the internal transaction when an error has been identified by the CLASS. When an error occurs and an error bit is set in the C0ISR, the internal transaction address is stored and the C0EARx is locked and does not update even if another error with a different transactions address occurs. Only when the AEIx bit in the C0ISR is cleared (either by a hardware reset or by writing a 1 to it) is C0EARx unlocked.

Table 4-6 lists the C0EARx bit field descriptions.

Table 4-6. C0EARx Bit Descriptions

Name	Reset	Description
ERR_ADD 31–0	0	**Error Address** This field stores the 32 lsbs of the address of the internal transaction that caused the error.

Note: The generated interrupts correspond to the following sources:

- C0EAR0 = Address generated by core 0.
- C0EAR1 = Address generated by core 1.
- C0EAR2 = Address generated by core 2.
- C0EAR3 = Address generated by core 3.
- C0EAR4 = Address generated by core 4.
- C0EAR5 = Address generated by core 5.
- C0EAR6 = Address generated by MAPLE-B port 0.
- C0EAR7 = Address generated by MAPLE-B port 1.
- C0EAR8 = Address generated by HSSI (serial RapidIO or PCI Express interface).
- C0EAR9 = Address generated by SEC, QUICC Engine subsystem (Ethernet or SPI), Debug system, or TDM.
- C0EAR10 = DMA port 0.
- C0EAR11 = DMA port 1.

4.8.5 CLASS Error Extended Address Registers (C0EEARx)

C0EEAR[0–11] CLASS Extended Error Address Registers Offset 0x9C0 + x*0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															RW
Type	R															
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
	—	SA	—					SRC_ID					ERR_ADD			
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0EEAR stores the most significant 4 bits of the address of the internal transaction when an error has been identified by the CLASS. This register also stores the attributes and the source ID of this transaction. When an error occurs and an error bit is set in the C0ISR, the internal transaction address is stored and the C0EEAR is locked and is not updated even if another error with a different transactions address/attributes occurs. Only when the AEI bit in the CISR is cleared (either by a hardware reset or by writing a 1 to it) is C0EEAR unlocked.

Table 4-7 lists the C0EEARx bit field descriptions.

Table 4-7. C0EEARx Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to 0 for future compatibility.	
RW 16	0	**Read/Write** This field indicates whether the transaction that caused the error was a read or a write.	0 Write. 1 Read.
— 15	0	Reserved. Write to 0 for future compatibility.	
SA 14	0	**Supervisor Access** This field indicates whether the transaction that caused the error was in supervisor mode.	0 Not supervisor. 1 Supervisor.
— 13–9	0	Reserved. Write to 0 for future compatibility.	
SRC_ID 8–4	0	**Source ID** Identifies the source ID of the initiator that caused the error.	0x00 Core 0 0x01 Core 1 0x02 Core 2 0x03 Core 3 0x06 MAPLE-B 0x07 MAPLE-B 0x08 DMA Controller 0x09 DMA Controller 0x0B QUICC Engine subsystem 0x0C Serial RapidIO Port 0 0x0D Serial RapidIO Port 1 0x0E TDM All other values reserved.
ERR_ADD 3–0	0	**Error Address** This field stores the 4 msbs of the address of the internal transaction that caused the error.	

4.8.6 CLASS Initiator Profiling Configuration Registers (C0IPCRx)

C0IPCR[0–11] CLASS Initiator Profiling Configuration Registers‘Offset 0xA00 + x*0x04

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

C0IPCRx controls the CLASS initiator profiling measurements. Each initiator has a dedicated C0IPCR which is numbered according to the initiator number within each CLASS module. The CLASS can perform only one measurement for a specific module at a time. Select the desired measurement for the initiator, enter the PMM value in the associated C0IPCR and make sure all the other C0IPCR and the C0TPCR for that CLASS are cleared.

Note: Only one PMM field among all C0IPCRx and C0TPCR can be greater than 0 during profiling.

Table 4-8 lists the C0IPCRx bit field descriptions.

Table 4-8. C0IPCRx Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to 0 for future compatibility.	
PMM 4–0	0	**Profiling Measurement Mode** Determines the profiling measurement mode for the matching initiator. **Note:** This register can only be cleared by a hardware reset.	00000 No measurement. 00001 Initiator priority and auto-upgrade. 00010 Initiator access type. 00011 Initiator stall. 00100 Initiator priority upgrade. 00101 Initiator priority non-upgrade. 00110 Initiator supervisor. 00111 Initiator bandwidth. 01000–01111 reserved 10000 Target 0 bandwidth. 10001 Target 1 bandwidth. 10010 Target 2 bandwidth. 10011 Target 3 bandwidth. 10100 Target 4 bandwidth. 10101 Target 5 bandwidth. 10110 Target 6 bandwidth. 10111 Target 7 bandwidth. 11000–11111 reserved

Table 4-9. Initiator Numbers

Initiator Number	Initiator Module
0	DSP core subsystem 0
1	DSP core subsystem 1
2	DSP core subsystem 2
3	DSP core subsystem 3
4	DSP core subsystem 4
5	DSP core subsystem 5
6	MAPLE port 0
7	MAPLE port 1
8	HSSI
9	SEC, Ethernet, SPI, TDM, or Debug
10	DMA port 0
11	DMA port 1

4.8.7 CLASS Initiator Watch Point Control Registers (C0IWPCRx)

C0IWPCR[0–11] CLASS Initiator Watch Point Control Registers Offset 0xA40 + x*0x04

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

C0IWPCRx controls the Watch Point Unit operation for the associated initiator. The Watch Point Unit monitors a specific access defined by the C0WPCRx, C0WPACRx, C0WPEACRx, and C0WPAMR. Each initiator can be enabled/disabled to monitor the selected access. You can write to this register while there are open CLASS transactions.

Note: Only one WPEN field can be set among all C0IWPCRx and C0TWPCRx when snooping watch point events.

Table 4-10 lists the C0IWPCRx bit field descriptions.

Table 4-10. C0IWPCRx Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to 0 for future compatibility.	
WPEN 0	0	**Watch Point Enable** Enables/disables the auto-upgrade mechanism. **Note:** This bit can only be cleared by a hardware reset.	0 The watch point is disabled. 1 The watch point is enabled.

4.8.8 CLASS Arbitration Weight Registers (C0AWRx)

C0AWR[0–11] CLASS Arbitration Weight Registers Offset 0xA80 + x*0x04

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

The value in C0AWRx determines the arbitration weight for the associated initiator. An initiator with arbitration weight of W is allowed to initiate up to W+1 consecutive transactions.

Note: When another initiator requests for access with higher priority level, the CLASS Arbiter chooses the higher priority request instead of the weighted winner.

Table 4-11 lists the C0AWRx bit field descriptions.

Table 4-11. C0AWRx Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to 0 for future compatibility.	
WEIGHT 3–0	0	**Weight** Contains the arbitration weight assigned to the associated initiator. **Note:** This register can only be cleared by a hardware reset.	**Recommended Values:** 0011 Use for C0AWR0 through C0AWR9 0111 Use for C0AWR10 and C0AWR11

Using the reset values can result in poor initial system performance. **Table 4-11** lists the recommended initial arbitration weight settings to apply after reset to increase initial performance levels during application development. These are just initial recommendations and can be changed by the designer according to the application requirements. However, the recommended settings will yield much higher performance than using the hardware default values. As a general rule, these recommended settings select a weighted arbitration of 3 for cores, MAPLE-B, RapidIO controllers, and other peripheral controllers; these are controlled by C0AWR0 through C0AWR9. Use a weighted arbitration of 7 for DMA transactions, which are controlled by C0AWR10 and C0AWR11. Also, see **Table 4-29** for recommended initial settings for the CLASS Arbitration Control Register (C0ACR).

4.8.9 CLASS0 Start Address Decoder x (C0SADx)

C0SAD5 CLASS0 Start Address Decoders Offset 0xC00 +x*0x04
C0SAD6

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								SA35	SA34	SA33	SA32	SA31	SA30	SA29	SA28
Type	R/W															
Reset																
SAD5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
SAD6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SA27	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	SA15	SA14	SA13	SA12
Type	R/W															
Reset																
SAD5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0SADx configure the address decoding of CLASS toward the DDR controllers (C0SAD5 and C0SAD6). They contain the start address of the window assigned to the specific port.

Note: To ensure proper operation, never modify the contents of the register while the specific decoder is enabled. Always clear the associated CATDx[DEN] bit before changing the contents of C0SADx.

These registers are reset by a hardware reset only. **Table 4-25** lists the C0SADx bit field descriptions.

Table 4-12. C0SADx Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
SA[35–12] 23–0	Port 5 = 0x040000 (DDR1 start) Port 6 = 0x080000 (DDR2 start)	**Start Address 35–12** The 24 msb of the start address of the specified port window. The lsbs are all zeros.

Note: Never write to these registers when there are open transactions being handled by the CLASS to the specified target controlled by the register.

4.8.10 CLASS End Address Decoder x (C0EADx)

C0EAD5 CLASS End Address Decoders Offset 0xC40 + x*0x04
C0EAD6

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								EA35	EA34	EA33	EA32	EA31	EA30	EA29	EA28
Type	R/W															
Reset																
EAD5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
EAD6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EA27	EA26	EA25	EA24	EA23	EA22	EA21	EA20	EA19	EA18	EA17	EA16	EA15	EA14	EA13	EA12
Type	R/W															
Reset																
EAD5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EAD6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C0EADx configure the address decoding of CLASS toward the DDR controllers (C0EAD5 and C0EAD6). They contain the end address of the window assigned to the specific port.

Note: To ensure proper operation, never modify the contents of the register while the specific decoder is enabled. Always clear the associated CATDx[DEN] bit before changing the contents of C0EADx.

These registers are reset by a hardware reset only. **Table 4-25** lists the C0EADx bit field descriptions.

Table 4-13. C0EADx Bit Descriptions

Name	Reset	Description
— 31–24	0	Reserved. Write to 0 for future compatibility.
EA[35–12] 23–0	Port 5 = 0x05FFFF (DDR1 end) Port 6 = 0x09FFFF (DDR2 end)	**End Address 35–12** The 24 msb of the end address of the specified port window. The lsbs are all zeros. You must make sure that this value is greater than or equal to the start address for the same window. If the end address is equal to the start address, the window size is 4 Kbytes.

Note: Never write to these registers when there are open transactions being handled by the CLASS to the specified target controlled by the register.

4.8.11 CLASS Attributes Decoder x (C0ATDx)

C0ATD5 CLASS0 Attributes Decoders Offset 0xC80 + X*0x04
C0ATD6

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

C0ATDx controls the functionality of each specific decoder for CLASS toward the DDR controllers (C0ATD5 and C0ATD6). They contain the bit that enables/disables the specific port.

If a decoder is disabled, it never indicates a hit, even if the address corresponds to the decoder address window. In this case the CLASS treats the space as if it is not assigned to any port. However, any transaction that was acknowledged up to and including the cycle in which DEN is cleared continues normally until completed.

Note: To ensure proper operation, do not enable the specific decoder before the start and end addresses are specified in the associated C0SADx and C0EADx.

These registers are reset by a hardware reset only. **Table 4-25** lists the C0ATDx bit field descriptions.

Table 4-14. C0ATDx Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to 0 for future compatibility.	
DEN 0	1	**Decoder Enable** Enables/disables the specified decoder.	0 Disables the decoder. 1 Enables the decoder.

Note: Never write to these registers when there are open transactions being handled by the CLASS to the specified target controlled by the register.

4.8.12 CLASS IRQ Status Register (C0ISR)

C0ISR CLASS IRQ Status Register Offset 0xD80

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—				AEI11	AEI10	AEI9	AEI8	AEI7	AEI6	AEI5	AEI4	AEI3	AEI2	AEI1	AEI0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0ISR indicates when an event occurs that requires the generation of an interrupt. There is a dedicated bit for each initiator. An interrupt is generated only when a status bit is set and the corresponding bit in the IRQ Enable register is set. Bits are cleared by writing ones to them. Writing a zero has no effect.

Note: You can write to or read this register at any time. The register is reset by a hard or soft reset.

Table 4-15 lists the C0ISR bit field descriptions.

Table 4-15. C0ISR Bit Descriptions

Name	Reset	Description	Settings
— 31–12	0	Reserved. Write to 0 for future compatibility.	
AEI[11–0] 11–0	0	**Address Error Interrupt 11–0** A bit is set if for a received transaction request, it does not belong to any port address space or falls inside one of the error areas.	0 No error. 1 Error detected.

4.8.13 CLASS IRQ Enable Register (C0IER)

C0IER CLASS IRQ Enable Register Offset 0xDC0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—				AEIE11	AEIE10	AEIE9	AEIE8	AEIE7	AEIE6	AEIE5	AEIE4	AEIE3	AEIE2	AEIE1	AEIE0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0IER is used to enable/disable the generation of interrupts that have occurred. There is a dedicated bit for each initiator. If a C0IER bit is cleared the corresponding bit in the C0ISR is masked. This register is reset by the hardware reset only.

Table 4-16 lists the C0IER bit field descriptions.

Table 4-16. C0IER Bit Descriptions

Name	Reset	Description	Settings
— 31–12	0	Reserved. Write to 0 for future compatibility.	
AEIE[11–0] 11–0	0	**Address Error Interrupt Enable** Used to enable/disable the address error interrupt for an initiator.	0 Interrupt masked. 1 Interrupt enabled.

4.8.14 CLASS Target Profiling Configuration Register (C0TPCR)

C0TPCR CLASS Target Profiling Configuration Register Offset 0xE00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—			TT	—	TN			—						PMM	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0TPCR is used to control the CLASS target profiling measurements. Each CLASS module can perform only one measurement for a specific module at a time. Use the values of TT and TN to select the module. Use the PMM value to select the measurement. Only write the PMM value to this register when all the C0IPCRx are cleared.

Note: For each CLASS module, you can only monitor one transaction. Therefore, only one PMM field in C0IPCRx and C0TPCR can be greater than 0 during profiling.

Table 4-17 lists the C0TPCR bit field descriptions.

Table 4-17. C0TPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–13	0	Reserved. Write to 0 for future compatibility.	
TT 12	0	**Target Type** Selects the module used for target profiling. Used with PMM. See PMM settings.	0 Arbiter. 1 Normalizer.
— 11	0	Reserved. Write to 0 for future compatibility.	
TN 10–8	0	**Target Number** Indicates the number of selected target.	000 CCSR 001 MAPLE 010 Core 4 and 5 Bridge 011 Core 2 and 3 Bridge 100 Core 0 and 1 Bridge 101 DDR1 110 DDR2 111 M3

Table 4-17. C0TPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 7–2	0	Reserved. Write to 0 for future compatibility.	
PMM 1–0	0	**Profiling Measurement Mode** Selects the profiling measurement for the selected target.	If TT = 0: 00 No profiling measurement. 01 Arbitration winner priority measurement. 10 Collisions measurement. 11 reserved. If TT = 1: 00 No profiling measurement. 01 Transaction splitting measurement. 10 Bandwidth measurement. 11 Stall measurement.

4.8.15 CLASS Profiling Control Register (C0PCR)

C0PCR CLASS Profiling Control Register Offset 0xE04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—						WPEC		—			TOE	—			PE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0PCR controls the CLASS profiling operation. The register is reset only by a hardware reset. **Table 4-18** lists the C0PCR bit field descriptions.

Table 4-18. C0PCR Bit Descriptions

Name	Reset	Description	Settings
— 31–10	0	Reserved. Write to 0 for future compatibility.	
WPEC 9–8	0	**Watch Point Event Configuration** Controls the effects of a Watch Point Unit event.	00 No effect. 01 Assertion of watch point event sets PE. 10 Assertion of watch point event clears PE. 11 Assertion of watch point event toggles PE.
— 7–5	0	Reserved. Write to 0 for future compatibility.	
TOE 4	0	**Time-Out Enable** Enables/disables the time-out mechanism.	0 Time-out function disabled. 1 Time-out function enabled.
— 3–1	0	Reserved. Write to 0 for future compatibility.	
PE 0	0	**Profiling Enable** Enables/disables the debug profiling unit operation.	0 Profiling unit disabled. 1 Profiling unit enabled.

4.8.16 CLASS Watch Point Control Registers (C0WPCR)

C0WPCR CLASS Watch Point Control Registers Offset 0xE08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															UPE
Type	R/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
	WCE	EATE	—	SIE	PRE	BCE	ATRE	ATAE	—				SPVE	RWE	AE	CE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0WPCR controls the CLASS watch point unit operation. You can configure this register to monitor a selected access type and count the number of times it occurs. The register is reset only by a hardware reset. **Table 4-19** lists the C0WPCR bit field descriptions.

Table 4-19. C0WPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to 0 for future compatibility.	
UPE 16	0	**Upgradeable Compare Enable** Enables/disables the time-out mechanism.	0 Upgradeable type compare disabled. 1 Upgradeable type compare with C0WPEACR enabled.
WCE 15	0	**Write-with-Confirm Compare Enable** Enables/disables the write-with-confirm type comparison.	0 Write-with-confirm type compare disabled. 1 Write-with-confirm type compare with C0WPEACR enabled.
EATE 14	0	**EOT Attributes Compare Enable** Enables/disables the EOT attributes comparison.	0 EOT attributes compare disabled. 1 EOT attributes compare with C0WPEACR enabled.
— 13	0	Reserved. Write to 0 for future compatibility.	
SIE 12	0	**Source ID Compare Enable** Enables/disables the source ID comparison.	0 Source ID compare disabled. 1 Source ID compare with C0WPEACR enabled.
PRE 11	0	**Priority Level Compare Enable** Enables/disables the priority level comparison.	0 Priority level compare disabled. 1 Priority level compare with C0WPEACR enabled.
BCE 10	0	**Byte Count Compare Enable** Enables/disables the byte count field comparison.	0 Byte count compare disabled. 1 Byte count compare with the field in C0WPEACR enabled.
ATRE 9	0	**Atomic Result Compare Enable** Enables/disables the atomic result type comparison.	0 Atomic result type compare disabled. 1 Atomic result type compare with C0WPACR enabled.
ATAE 8	0	**Atomic Access Compare Enable** Enables/disables the atomic access type comparison.	0 Atomic access type compare disabled. 1 Atomic access type compare with C0WPACR enabled.
— 7–4	0	Reserved. Write to 0 for future compatibility.	
SPVE 3	0	**Supervisor Access Compare Enable** Enables/disables supervisor access comparison.	0 Supervisor type compare disabled. 1 Supervisor type compare with C0WRACR enabled.

Table 4-19. C0WPCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RWE 2	0	**Read/Write Compare Enable** Enables/disables read/write type comparison.	0 Read/write type compare disabled. 1 Read/write type compare with C0WPACR enabled.
AE 1	0	**Address Compare Enable** Enables/disables comparison of the access address.	0 Address compare disabled. 1 Address compare with C0WPACR enabled.
CE 0	0	**Count Enable** Enables/disables the counter for watch point events.	0 Counter 1 disabled for watch point events. 1 Counter 1 enabled for watch point events.

4.8.17 CLASS Watch Point Access Configuration Register (C0WPACR)

C0WPACR CLASS Watch Point Access Configuration Registers Offset 0xE0C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ATR	ATA	—				SPV	RW	ADDR							
Type	R/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
	ADDR															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0WPACR, along with C0WPEACR, configures the selected access to monitor. The watch point monitoring occurs only if the respective function is enabled in the C0WPCR. The register is reset only by a hardware reset. **Table 4-20** lists the C0WPACR bit field descriptions.

Table 4-20. C0WPACR Bit Descriptions

Name	Reset	Description	Settings
ATR 31	0	**Atomic Result** Defines the atomic result type to monitor.	0 Atomic access failed. 1 Atomic access succeeded.
ATA 30	0	**Atomic Access** Defines the atomic access type to monitor.	0 Non-atomic access. 1 Atomic access.
— 29–26	0	Reserved. Write to 0 for future compatibility.	
SPV 25	0	**Supervisor Access** Defines the supervisor access type to monitor.	0 Non-supervisor access. 1 Supervisor access.
RW 24	0	**Read/Write Access** Defines the access type to monitor.	0 Write. 1 Read.
ADDR 23–0	0	**Address[35–12]** This field, along with the ADDM field in C0WPAWMR, defines the start and range of the addresses the watch point unit monitors. **Note:** For every bit in C0WPAMR[ADDM] that is cleared, make sure the corresponding bit is cleared in the ADDR. The bit location in ADDM (b) corresponds to the b + 12 bit location in ADDR.	

4.8.18 CLASS Watch Point Extended Access Configuration Register (C0WPEACR)

C0WPEACR CLASS Watch Point Extended Access Configuration Registers Offset 0xE10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	UP	WC	—		EATTR				—							
Type	R/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
	SI					PR		BC								
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0WPEACR, along with C0WPACR, configures the selected access to monitor. The watch point monitoring occurs only if the respective function is enabled in the C0WPCR. The register is reset only by a hardware reset. **Table 4-21** lists the C0WPEACR bit field descriptions.

Table 4-21. C0WPEACR Bit Descriptions

Name	Reset	Description	Settings
UP 31	0	**Upgradeable Access** Defines the upgradeable access type to monitor.	0 Non-upgradeable access. 1 Upgradeable access.
WC 30	0	**Write-with-Confirm Access** Defines the write-with-confirm access type to monitor.	0 Fast confirm access. 1 Write-with-confirm access.
— 29–28	0	Reserved. Write to 0 for future compatibility.	
EATTR 27–24	0	**EOT Attributes** Defines the EOT attributes to monitor.	0x0– 0x7 Reserved 0x8 Target port 0 CCSRs 0x9 Target port 1 MAPLE-B 0xA Target port 2 Reserved 0xB Target port 3 Cores 2 and 3 0xC‘ Target port 4 Cores 0 and 1 0xD Target port 5 DDRC1 memory 0xE Target port 6 DDRC2 memory 0xF Target port 7 M3 memory
— 23–16	0	Reserved. Write to 0 for future compatibility.	

Table 4-21. C0WPEACR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SI 15–11	0	**Source** Defines the source ID to monitor.	0x00 Core0 0x01 Core1 0x02 Core2 0x03 Core3 0x04 Core 4 0x05 Core 5 0x06 MAPLE Port 0 0x07 MAPLE Port 1 0x08 SerDes Bridge 0x09 Peripherals Bridge 0x0A DMA Port 0 0x0B DMA Port 1
PR 10–9	0	**Priority** Defines the priority level to monitor.	00 Priority 0 (highest) 01 Priority 1 10 Priority 2 11 Priority 3 (lowest)
BC 8–0	0	**Byte Count** This field defines the value of the byte count that the watch point unit monitors.	The byte count to monitor can be from 1 to 511 bytes.

4.8.19 CLASS Watch Point Address Mask Registers (C0WPAMR)

C0WPAMR CLASS Watch Point Address Mask Registers Offset 0xE14

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

C0WPAMR controls the address range monitored by the watch point unit. The register is reset only by a hardware reset. **Table 4-22** lists the C0WPAMR bit field descriptions.

Table 4-22. C0WPAMR Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to 0 for future compatibility.	
ADDM 7–0	0	**Address Mask** Defines the range and alignment of the address to monitor if address monitoring is enabled. The start address is defined in C0WPACR[ADDR]. **Note:** For every bit in ADDM that is cleared, make sure the corresponding bit is cleared in the C0WPACR.	00000000 Aligned with a range of 1 MB. 10000000 Aligned with a range of 512 KB. 11000000 Aligned with a range of 256 KB. 11100000 Aligned with a range of 128 KB. 11110000 Aligned with a range of 64 KB. 11111000 Aligned with a range of 32 KB. 11111100 Aligned with a range of 16 KB. 11111110 Aligned with a range of 8 KB. 11111111 Aligned with a range of 4 KB. All other values are reserved.

4.8.20 CLASS Profiling Time-Out Registers (C0PTOR)

C0PTOR CLASS Profiling Time-Out Registers Offset 0xE18

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

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

C0PTOR is used to stop the profiling unit operation. When the C0PRCR reaches the value stored in C0PTOR and C0PCR[TOE] is set, the CLASS clears the C0PCR[PE] bit to disable the profiling unit. When C0PCR[PE] clears, the CLASS stops all profiling counters. The register is reset only by a hardware reset. **Table 4-23** lists the C0PTOR bit field descriptions.

Table 4-23. C0PTOR Bit Descriptions

Name	Reset	Description
TO 31–0	0xFFFFFFFF	**Time-Out** Holds the time-out value used to stop the profiling unit when the time-out function is enabled.

4.8.21 CLASS Target Watch Point Control Registers (C0TWPCR)

C0TWPCR CLASS Target Watch Point Control Registers Offset 0xE1C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—								WPEN7	WPEN6	WPEN5	WPEN4	WPEN3	WPEN2	WPEN1	WPEN0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0TWPCR controls the watch point unit operation for CLASS targets. The watch point unit monitors a specific access defined in C0WPCR, C0WPACR, C0WPEACR, and C0WPAMR. Each target can be enabled/disabled for monitoring the specified access type. The register is reset by a hard reset only.

Note: Only one WPEN field can be set among all the C0IWPCRx and C0TWPCR. That is, only one watch point unit can be active at a time.

Table 4-15 lists the C0TWPCR bit field descriptions.

Table 4-24. C0TWPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to 0 for future compatibility.	
WPEN[7–0] 7–0	0	**Watch Point Enable 7–0** Each bit enables monitoring of access by the associated target.	0 The watch point unit for the associated target is disabled. 1 The watch point unit for the associated target is enabled.

4.8.22 CLASS Profiling IRQ Status Register (C0PISR)

C0PISR CLASS Profiling IRQ Status Registers Offset 0xE20

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

C0PISR indicates that a watch point event occurred or that the C0PRCR overflowed. An interrupt is generated if the status bit is set and the corresponding bit in C0PIERx is set to enable the interrupt. You can write to or read the register at any time. Write a 1 to a bit to clear it; writing a 0 has no effect. The register is reset by a hard or soft reset. **Table 4-25** lists the C0PISR bit field descriptions.

Table 4-25. C0PISR Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to 0 for future compatibility.	
WPE **1**	0	**Watch Point Event** Enables monitoring of access by the associated target.	0 No watch point event occurred. 1 Watch point event captured.
OVE **0**	0	**Overflow Event** Enables monitoring of access by the associated target.	0 No overflow occurred. 1 C0PRCR overflowed (reached 0xFFFFFFFF) during the last measurement.

4.8.23 CLASS Profiling IRQ Enable Register (C0PIER)

C0PIER CLASS Profiling IRQ Enable Registers Offset 0xE24

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

C0PIER enables/disables the generation of interrupts by the debug profiling unit. You can write to the register at any time. The register is reset by a hard reset only. **Table 4-26** lists the C0PIER bit field descriptions.

Table 4-26. C0PIER Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to 0 for future compatibility.	
WPEE **1**	0	**Watch Point Event Enable** Enables/disables a watch point interrupt.	0 Watch point interrupt is masked. 1 Watch point interrupt is enabled.
OVEE **0**	0	**Overflow Event Enable** Enables/disables an overflow interrupt.	0 Overflow interrupt is masked. 1 Overflow interrupt is enabled.

4.8.24 CLASS Profiling Reference Counter Register (C0PRCR)

C0PRCR CLASS Profiling Reference Counter Registers Offset 0xE40

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	C0T															
Type	R															
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
	CNT															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C0PRCR is the reference counter for all profiling measurements. This read-only register counts the number of cycles occurring during the profiling measurement or during the watch point unit operation. The counter starts counting from zero when the profiling unit is enabled. The C0PRCR stops when the profiling unit is disabled, or when the C0PRCR reaches the value stored in C0PTOR and TOE is set, which causes the CLASS to clear the PE bit to disable the profiling

unit. When PE clears, the CLASS stops all profiling counters. The register is reset only by a hardware reset only. **Table 4-27** lists the C0PRCR bit field descriptions.

Table 4-27. C0PRCR Bit Descriptions

Name	Reset	Description
CNT 31–0	0	**Counter** Holds the reference counter for the profilers.

4.8.25 CLASS Profiling General Counter Registers (C0PGCRx)

C0PGCR[0–3] CLASS Profiling General Counter Registers Offset 0xE44 + x*0x04

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

C0PGCRx is used to count profiling unit or watch point unit events. This read-only register counts the number of cycles occurring during the profiling measurement or during the watch point unit operation. The counter starts counting from zero when the profiling unit is enabled. The C0PRCR stops when the profiling unit is disabled, or when the C0PRCR reaches the value stored in C0PTOR and TOE is set, which causes the CLASS to clear the PE bit to disable the profiling unit. When PE clears, the CLASS stops all profiling counters. The register is reset only by a hardware reset. **Table 4-28** lists the C0PGCR bit field descriptions.

Table 4-28. C0PGCR Bit Descriptions

Name	Reset	Description
CNT 31–0	0	**Counter** Holds the counter value of the selected measurement. **Table 4-2** lists the measurements counted by each counter for each configuration combination.

4.8.26 CLASS Arbitration Control Register (C0ACR)

C0ACR CLASS Arbitration Control Registers Offset 0xFC0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			PME	—											
Type	R/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
	—								LA7	LA6	LA5	LA4	LA3	LA2	LA1	LA0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0ACR controls the CLASS arbiters. There is a dedicated bit for each arbiter that controls the Late Arbitration mode of the associated arbiter. When Late Arbitration mode is enabled, the arbiter delays the decision according to the size of the last winner access. The arbiter calculates the number of cycles to allow until a winner decision must be made. This is done to keep the bus always full, and make the winner decision as late as possible. When Late Arbitration mode is disabled, the arbiter makes a decision every clock cycle.The register is reset by a hard reset only. **Table 4-29** lists the C0ACR bit field descriptions.

Table 4-29. C0ACR Bit Descriptions

Name	Reset	Description	Settings
— 31–29	0	Reserved. Write to 0 for future compatibility.	
PME **28**	0	**Priority Mask Enable** Enables/disables the operation of the priority mask unit for starvation elimination.	0 Priority mask disabled. 1 Priority mask enabled.
— 27–8	0	Reserved. Write to 0 for future compatibility.	
LA[7–0] 7–0	0	**Late Arbitration 7–0** Enables/disables late arbitration mode for the associated arbiter. **Note:** As with the arbitration weight (see **Section 4.8.8**, *CLASS Arbitration Weight Registers (C0AWRx)*, on page 4-26), the default value for this field does not yield optimal performance. Freescale recommends that after reset, write a value of 0xFC to this field. This value assigns late arbitration to the cores, the M2 memory, DDR memory, and the M3 memory. This is just an initial value, and can be changed according to the application requirements and system traffic.	0 Late arbitration disabled. 1 Late arbitration enabled.

Reset 5

The reset and control signals provide many options for MSC8156E operation by configuring various modes and features during power-on reset. Most of these features are configured by loading a reset configuration word to the MSC8156E device that combine with a few direct configuration inputs sampled during the reset sequence. This section describes the various ways to reset and configure the MSC8156E device.

5.1 Reset Operations

The MSC8156E has several inputs to the reset logic:

- Power-on reset ($\overline{\text{PORESET}}$)
- External hard reset ($\overline{\text{HRESET}}$)
- External soft reset ($\overline{\text{SRESET}}$)
- Software watchdog reset
- JTAG reset
- RapidIO reset
- Software hard reset
- Software soft reset

All of these reset sources are fed into the reset controller and, depending on the source of the reset, different actions are taken. The reset status register described in **Section 5.3.3** indicates the last sources to cause a reset.

5.1.1 Reset Sources

Table 5-1 describes reset sources.

Table 5-1. Reset Sources

Name	Description
Power-on reset (PORESET)	Input pin. Asserting this pin initiates the power-on reset flow that resets all the device and configures various attributes of the device including its clock modes.
Hard reset (HRESET)	This is a bidirectional I/O pin. The MSC8156E can detect an external assertion of HRESET only if it occurs while the MSC8156E is not asserting reset. During HRESET, SRESET is asserted. HRESET is an open-drain pin.
Soft reset (SRESET)	Bidirectional I/O pin. The MSC8156E can only detect an external assertion of SRESET if it occurs while the MSC8156E is not asserting reset. SRESET is an open-drain pin.
Software watchdog reset	After the MSC8156E watchdog timer counts to zero, a software watchdog reset is generated. The enabled software watchdog event then generates an internal hard reset sequence.
RapidIO reset	When the RapidIO logic asserts the RapidIO hard reset signal, an internal hard reset sequence is generated.
JTAG reset	When JTAG logic asserts the JTAG soft reset signal, an internal soft reset sequence is generated.
Software hard reset	A hard reset sequence can be initialized by writing to a memory mapped register (RCR)
Software soft reset	A soft reset sequence can be initialized by writing to a memory mapped register (RCR)

5.1.2 Reset Actions

The MSC8156E reset control logic determines the cause of reset, synchronizes it if necessary, and resets the appropriate internal hardware. Each reset flow has different impact on the device logic. Power-on reset has the greatest impact, resetting the entire device, including clock logic and error capture registers. Hard reset resets the entire device excluding clock logic and error capture registers, while Soft reset initializes the internal logic while maintaining the system configuration. All reset types generate a reset to the cores. The memory controllers, system protection logic, interrupt controller, and I/O pins are initialized only on hard reset. Soft reset initializes the internal logic while maintaining the system configuration. Asserting external SRESET generates a soft reset to the DSP cores and to the remainder of the device. **Table 5-2** identifies reset actions for each reset source.

Table 5-2. Reset Actions for Each Reset Source

Reset Source	Clocks and PLLs Reset Logic Error Capture Registers	Performance Monitor HSSI PLLs and Logic Timers CLASS (most registers, see Section 4.7, *Programming Model*, on page 4-11 for details)	Reset Configuration Words Loaded	Other Internal Logic	$\overline{\text{HRESET}}$ Driven	$\overline{\text{SRESET}}$ and Soft Reset Driven to Cores
• Power-on reset	Yes	Yes	Yes	Yes	Yes	Yes
• External hard reset • Software watchdog reset • RapidIO reset • Software hard reset	No	Yes	No	Yes	Yes	Yes
• External soft reset • JTAG soft reset • Software soft reset	No	No	No	Yes	No	Yes

5.1.3 Power-On Reset Flow

Assertion of the external $\overline{\text{PORESET}}$ signal initiates the power-on reset flow. $\overline{\text{PORESET}}$ should be asserted externally for at least 32 input clock cycles after stable external power is applied to the MSC8156E device. When $\overline{\text{PORESET}}$ is deasserted, the MSC8156E starts the configuration process. The MSC8156E asserts $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the power-on reset sequence, including during configuration. Configuration time varies according to the configuration source and CLKIN frequency. Initially, the reset configuration inputs are sampled to determine the configuration source and the input clock division mode. Next, the MSC8156E starts loading the reset configuration words. When the clock mode values in the reset configuration word low load, the PLLs begin to lock, after locking, each PLL distributes clock signals to the device. When all clocks are locked and the reset configuration words are loaded, $\overline{\text{HRESET}}$ is released. $\overline{\text{SRESET}}$ is released 21 or 36 clocks later (depending on the reset configuration word source).

5.1.4 Detailed Power-On Reset Flow

The detailed power-on reset ($\overline{\text{PORESET}}$) flow for the MSC8156E is as follows:

1. The user asserts $\overline{\text{PORESET}}$ (and optionally $\overline{\text{HRESET}}$).
2. Power is applied to meet the specifications in the *MSC8156E Technical Data Sheet*.
3. The user asserts $\overline{\text{PORESET}}$ (and optionally $\overline{\text{HRESET}}$) causing all registers to be initialized to their default states and most I/O drivers to be tri-stated (some clock, clock enabled, and system control signals are active).
4. The user applies a stable CLKIN signal and stable reset configuration inputs (RCW_SRC, RC, STOP_BS).
5. Deassert $\overline{\text{PORESET}}$ after at least 32 stable CLKIN clock cycles; counting the 32 cycles should only start after V_{DDIO} has reached its nominal value as specified in the *MSC8156E Technical Data Sheet*.
6. The device samples the reset configuration input signals to determine the reset configuration word source.
7. The device starts loading the reset configuration word. Loading time depends on the reset configuration word source.
8. Once Reset Configuration Word Low is loaded, the system PLL begins to lock.
9. The device keeps driving $\overline{\text{HRESET}}$ low until all PLLs are locked and the reset configuration words are loaded.
10. Deassert the optional $\overline{\text{HRESET}}$, if not done earlier.

Note: The JTAG logic must always be initialized by asserting $\overline{\text{TRST}}$. There is no need to assert $\overline{\text{SRESET}}$ when $\overline{\text{HRESET}}$ is asserted.

11. The internal reset to the cores and the remaining logic is deasserted. I/O drivers are enabled.
12. After $\overline{\text{HRESET}}$ is deasserted, it can take 41000 OCN cycles for the SerDes block to exit reset and lock its internal PLL. Read the HSSI_SR[SERDES1_RST_DONE] and HSSI_SR[SERDES2_RST_DONE] bits to determine when the SerDes reset is complete. See the High Speed Serial Interface Status Register (HSSI_SR] description in **Chapter 8** *General Configuration Registers* for details.
13. If enabled, the HSSI complex interfaces are now ready to accept external requests, and the core boot code fetch can proceed, if enabled. The MSC8156E is now in its ready state.

Figure 5-1 shows a timing diagram of the power-on reset flow.

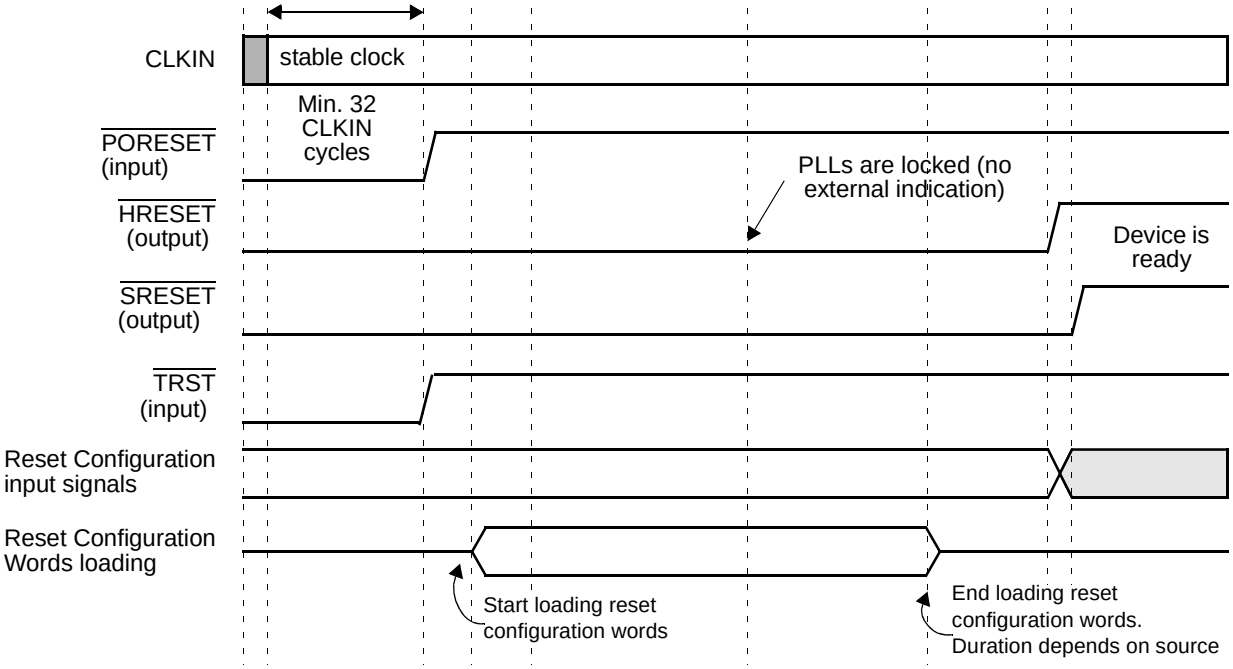


Figure 5-1. Power-On Reset Flow

Figure 5-2 shows a timing diagram of power-on reset flow with RCW_SRC = 000. In this mode, the external pins RC[15–0] are sampled four times in order to get all the 64 bits RCW.

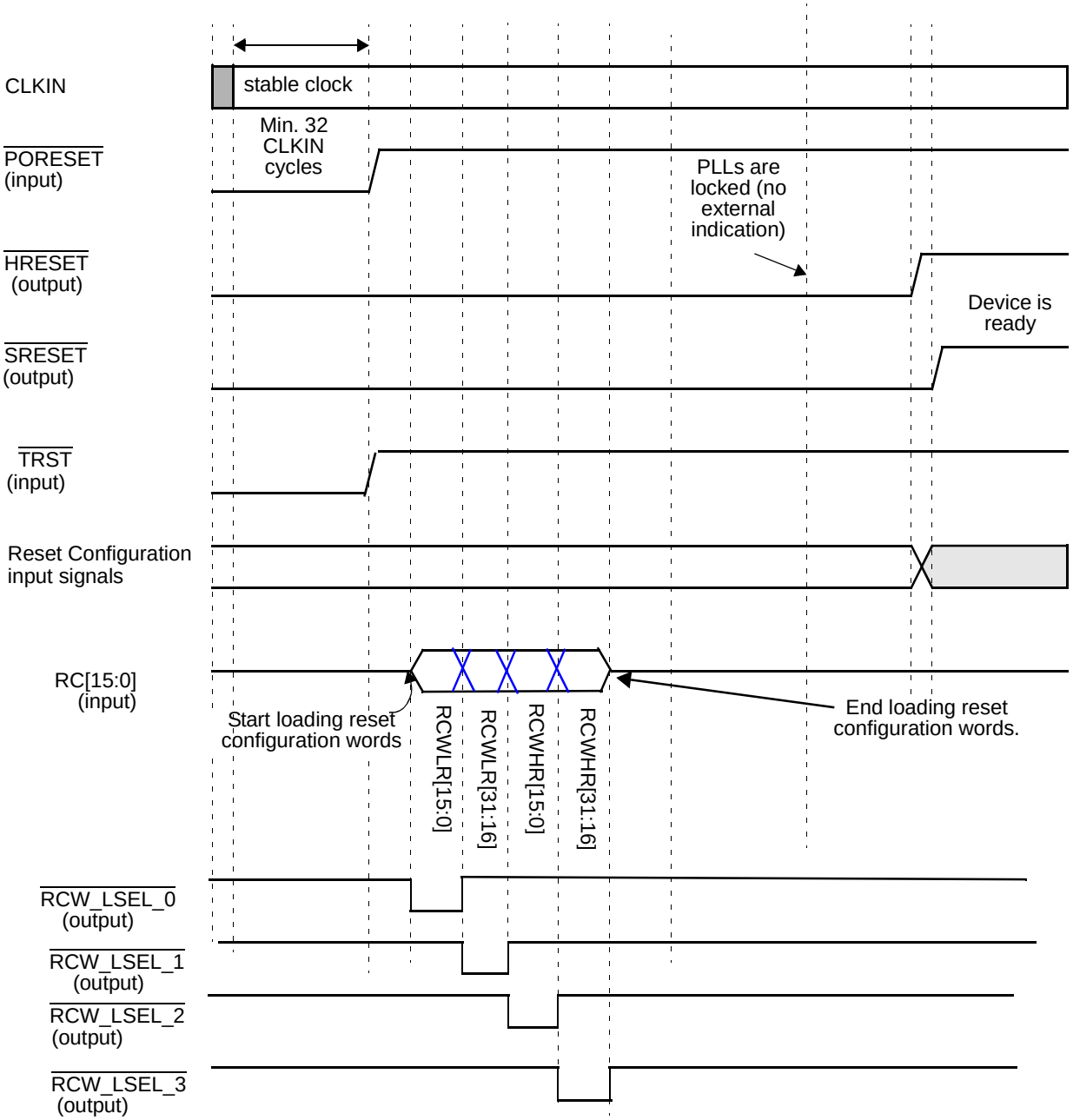


Figure 5-2. Power-on Reset Flow for RCW_SRC = 000

5.1.5 HRESET Flow

The HRESET flow may be initiated externally by asserting HRESET or internally when the MSC8156E detects a reason to assert HRESET. In both cases, the device continues asserting HRESET and SRESET throughout the HRESET flow. The hard reset sequence time is 1286 CLKIN cycles. The reset configuration source, the reset configuration word, and the input clock division mode are not affected by hard reset (they are only affected by a power-on reset), so the MSC8156E immediately configures the device. After the configuration sequence completes, the MSC8156E releases both HRESET and SRESET signals and exits the HRESET flow. Use an external pull-up resistor to deassert the signals. After deassertion is detected, the device waits for a 16-cycle period before testing the presence of an external (hard/soft) reset. The HSSI complex logic and PLL are out of reset after about 41000 OCN cycles. Read the HSSI_SR[SERDES1_RST_DONE] and HSSI_SR[SERDES2_RST_DONE] bits to determine when the SerDes reset is complete. See the High Speed Serial Interface Status Register (HSSI_SR] description in **Chapter 8** *General Configuration Registers* for details.

Note: Because the MSC8156E does not sample the reset configuration signals (RCW_SRC, RC, STOP_BS) during a hard reset flow, setting a new value on those pins (different from the settings during power-on reset) has no effect.

Figure 5-3 shows a timing diagram of the hard reset flow.

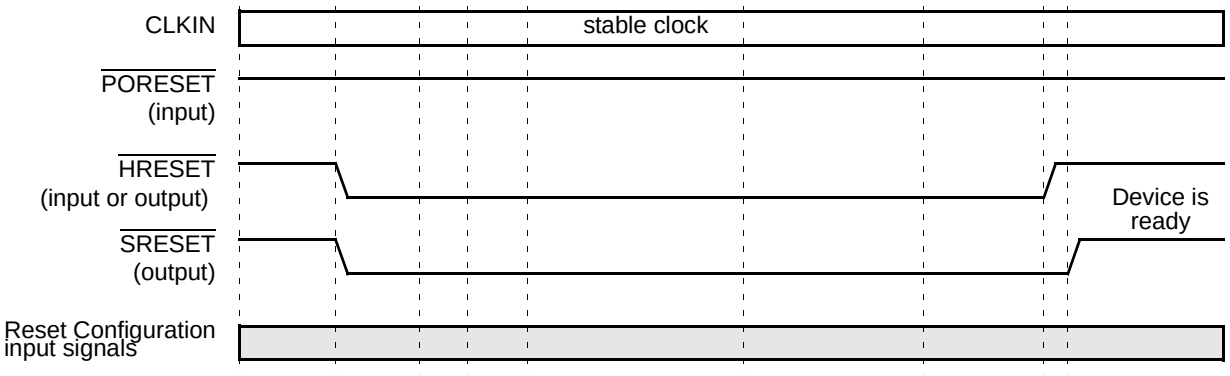


Figure 5-3. Hard Reset Flow

5.1.6 SRESET Flow

The SRESET flow is initiated externally by asserting SRESET or internally when the MSC8156E detects a cause to assert SRESET. In both cases, the MSC8156E asserts SRESET for 512 CLKIN clock cycles, after which the MSC8156E releases SRESET and exits the SRESET state. An external pull-up resistor should be used to deassert SRESET; after deassertion is detected, the device waits for a 16-cycle period before testing for the presence of an external (hard/soft) reset. When SRESET is asserted, internal hardware is reset, but the hard reset configuration does not change.

5.2 Reset Configuration

The MSC8156E is initialized using two complementary methods. Initially, a small number of input signals (RCW_SRC[0–2]) are sampled during the first two CLKIN cycles after the deassertion of $\overline{\text{PORESET}}$ (during the power-on reset flow). These signals determine whether a reset configuration word is required and the device source interface from which it is loaded (see **Table 5-1**). The RCW_SRC[0–2] signals should remain valid until the deassertion of $\overline{\text{HRESET}}$. Depending on the configuration signal values, the MSC8156E may continue with loading the reset configuration word.

5.2.1 Reset Configuration Signals

Reset configuration input signals are located on device pins that have other functions when the device is not in the reset state. These input signals sampled values are written into registers during the assertion of $\overline{\text{PORESET}}$ after a stable clock is supplied (the power-on reset flow). The inputs must be pulled high or low by external resistors as long as $\overline{\text{HRESET}}$ is asserted. During the $\overline{\text{PORESET}}$ flow, all other signal drivers connected to these signals must be tri-stated. Refer to the *MSC8156E Technical Data Sheet* for the recommended resistor values used to pull reset configuration signals high or low. The values loaded from these sampled inputs are accessible to software through memory-mapped registers described in **Section 5.3**. They are used to configure the device operation.

5.2.2 Reset Configuration Words Source

The reset configuration word source options permit the MSC8156E to load reset configuration words from an EEPROM via the I^2C interface, a combination of external pins and hard-coded values, or to use hard-coded default options.

Table 5-3. Reset Configuration Word Sources (Defined by RCW_SRC[0–2])

Value (Binary)	Meaning
000	Multiplexed external RCW loading. The RCW is driven by external logic on RC[15–0]. The $\overline{\text{RCW\_LSEL}}$[3–0] selects which bits should be driven on RC[15–0].
001	Reset configuration word is loaded from an I^2C EEPROM in 8-bit addressing mode. The internal MSC8156E hardware reads the RCW from EEPROM slave address 1010000 (or 1010111 in case of multi device and reset slave) with single byte addressing.
010	Reset configuration word is loaded from an I^2C EEPROM in 16-bit addressing mode. MSC8156E hardware reads the RCW from EEPROM slave address 1010000 (or 1010111 in case of multi device and reset slave) with two byte addressing.
011	Some bits of the reset configuration word are loaded from external pins and others by default.
100	Hard coded option #1. Reset configuration word is loaded from internal hard coded option 1.
101	Hard coded option #2. Reset configuration word is loaded from internal hard coded option 2.

Note: The value of the reset configuration signals affects the duration of power-on and hard reset sequences.

5.2.3 Reset Configuration Input Signal Selection and Reset Sequence Duration

Table 5-4 shows how to pull down (0) or pull up (1) the reset configuration input signals (RCW_SRC) for various configurations. The reset sequence duration is measured from the deassertion of $\overline{\text{PORESET}}$ to the deassertion of $\overline{\text{HRESET}}$. The $\overline{\text{SRESET}}$ signal is deasserted 21 CLKIN cycles after the deassertion of $\overline{\text{HRESET}}$ for sources that do not use the I^2C interface and 36 CLKIN cycles for sources that use the I^2C interface.

Table 5-4. Selecting Reset Configuration Input Signals

CLKIN Frequency	RCW_SRC[0–2]	Reset Sequence Duration in CLKIN Cycles	Duration in ms
66 MHz	000, 011–101	17426	0.264
66 MHz	001, 010	255156	3.866
100 MHz	000, 011-101	17426	0.174
100 MHz	001, 010	255156	2.552

5.2.4 Reset Configuration Words

Various device functions are initialized by loading the reset configuration words during the power-on reset flow. All configurable features are reconfigured only during a power-on reset flow. The MSC8156E decides which interface is used according to reset configuration input signals, as described in **Section 5.2.2**.

Section 5.3 describes the functions and modes configured by the reset configuration words. Note that the reset configuration settings are accessible to software through the following read-only memory-mapped registers:

- Reset Configuration Word Low Register (RCWLR). See **Section 5.3.1** for details.
- Reset Configuration Word High Register (RCWHR). See **Section 5.3.2** for details.
- Reset Status Register (RSR). See **Section 5.3.3** for details.

5.2.5 Loading The Reset Configuration Words

The MSC8156E loads the reset configuration words from an I^2C serial EEPROM, or combination of default values and external pins, or uses a hard-coded configuration, as selected by the reset configuration inputs described in **Section 5.2.2**. The following subsections describe these options in detail.

5.2.5.1 Loading From an I2C EEPROM (RCW_SRC[0–2] = 001 or 010)

When a MSC8156E is configured by the reset configuration input signals to load the reset configuration words from an EEPROM via the I2C interface, it uses the I2C unit boot sequencer in a special mode. In this mode, the I2C boot sequencer is activated to load the reset configuration words while the rest of the device remains in the reset state ($\overline{\text{HRESET}}$ is asserted).

5.2.5.1.1 Using The Boot Sequencer For Reset Configuration

Note: For detailed description about the I2C interface and the boot sequencer, refer to **Chapter 26**, *I2C*.

When used to load the reset configuration words, the I2C module addresses the first EEPROM, reads the preamble, and then reads the first two data structures. The device latches the reset configuration words internally and the I2C module enters its reset state until $\overline{\text{HRESET}}$ is deasserted. There should be no other I2C traffic when the boot sequencer is active. After $\overline{\text{HRESET}}$ is deasserted, the boot sequencer mode is disabled.

5.2.5.1.2 EEPROM Slave Address

A reset master MSC8156E is selected by holding its STOP_BS signal low during the power-on reset flow. The reset master uses 0b1010000 for the EEPROM calling address. A reset slave uses 0b1010111 for the EEPROM calling address. The EEPROM to be addressed must contain the reset configuration information and be programmed to respond to the 0b1010000 address. The EEPROM device must have address inputs connected to GND in multi device reset applications. No additional EEPROMs are accessed by the boot sequencer in reset configuration mode. See also **Section 5.2.5.1.5**, *Loading Multiple Devices From a Single I2C EEPROM*, on page 5-11.

5.2.5.1.3 EEPROM Data Format In Reset Configuration Mode

The I2C module expects a specific data format in the EEPROM. The first three bytes should be the preamble and should contain a value of 0xAA55AA. The I2C module verifies that this preamble is correctly detected before proceeding further. The two reset configuration words, should follow the preamble and should use the required format provided in **Section 6.2.2.3**, *Boot File Format*, on page 6-16. Within each configuration word, the first 3 bytes are reserved and must contain the value 0xFFFFFF. After the first 3 bytes, 4 bytes of data should hold the desired value of the reset configuration word. The boot sequencer assumes that a big endian address is stored in the EEPROM.

If a preamble fail or any other I2C bus error is detected, the device stops processing and remains in a hard reset state with $\overline{\text{HRESET}}$ asserted and most of the I/O drivers are disabled. If reset configuration word loading from the EEPROM fails, the user must assert the $\overline{\text{PORESET}}$ signal for at least 100 µs duration to resume operation.

5.2.5.1.4 Single Device Loading From I²C EEPROM

The MSC8156E can be the only device loading the reset configuration word from the I²C EEPROM. In this case STOP_BS pin must be driven low during the power on and hard reset sequences. The hardware connection is shown in **Figure 5-4**.

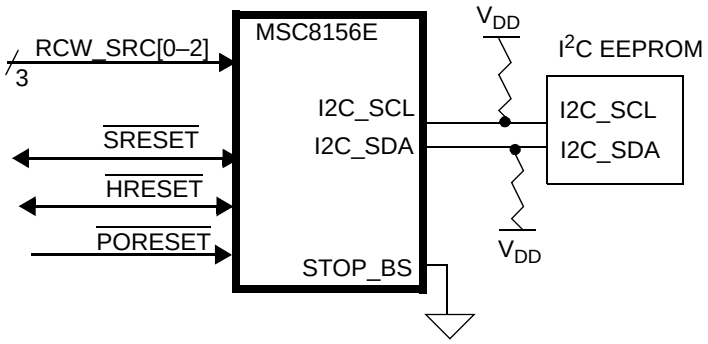


Figure 5-4. Single Device I²C Reset Configuration

5.2.5.1.5 Loading Multiple Devices From a Single I²C EEPROM

When the MSC8156E device shares the I²C EEPROM device with other MSC8156E devices to load the reset configuration words, one device must be a reset master and the rest must be reset slaves. The definition of reset slave or reset master is latched internally during power-on reset sequence. In this mode, the RCW_SRC inputs must be the same for all slaves and the master.

The reset configuration implementation involves a software for the reset master and glue logic. The hardware connection is shown in **Figure 5-5**. The STOP_BS signal input to the reset master must be driven low during the power on reset sequence while all the slaves inputs must be driven high. During the power on reset assertion, the master cannot drive the STOP_BS output bus because its role as master is not enabled yet. Pull-ups are required; refer to the *MSC8156E Technical Data Sheet* for appropriate resistor values to pull the slave STOP_BS input signal high.

In the first stage of reset configuration, the reset master reads its own reset configuration words. It accesses the I²C EEPROM while all other reset slaves are stopped. When PORESET is deasserted, the STOP_BS is latched in the reset block after few cycles and defines the reset master and slaves. It also keeps all the reset slave I²C controllers in idle state while the reset master starts to access the EEPROM slave using address 0b1010000. Then the reset master must exit from reset and run the internal code.

In the second stage, the reset master reads the slave RCWs and stores the values in its memory. See **Chapter 6**, *Boot Program* for how to determine the number of reset slaves to be configured. The reset master core reads the slave RCWs from the I²C EEPROM. Then, it configures its I²C controller to emulate an EEPROM device for each reset slave. The reset master emulates EEPROM using the slave address 0b1010111.

In the last stage, the reset master releases the STOP_BS for each slave in a known order. The released reset slave accesses the I^2C bus to read from slave address 0b1010111. The order of reading the slave RCWs is the order for their connection.

Note: The STOP_SLV_BS glue logic supports a five pin bus. For up to five reset slaves, the master drives STOP_SLV_BS directly to the selected slave. For five to fifteen slaves, the glue logic encodes the signals from the master and selects the slave to drive with STOP_SLV_BS based on its decoded value.

The external reset logic may reset the system as a unit, or it may be configured to reset individual devices. Individual resets permit redundancy support during system debugging to allow problematic devices to be disabled and replaced by a redundant device.

Multi device is described in detail in **Section 6.1.4**, *Multi Device Support for the I2C Bus*, on page 6-4.

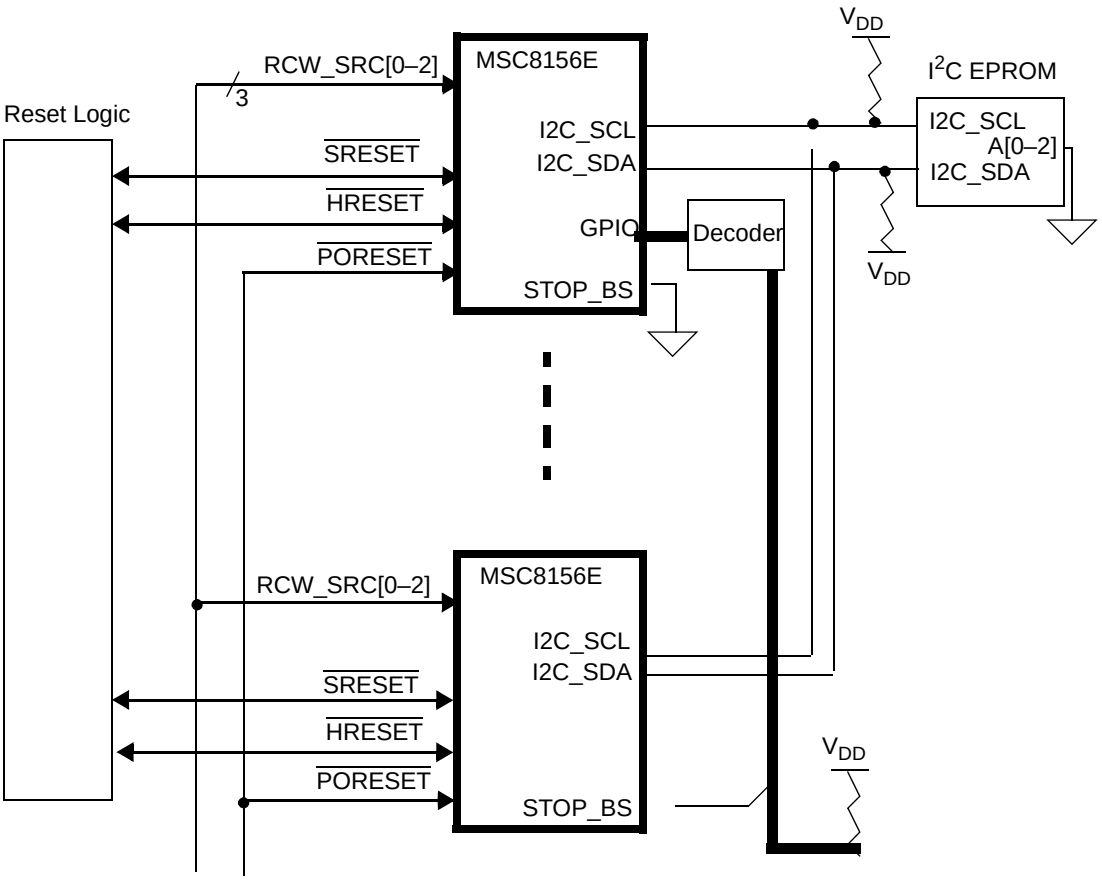


Figure 5-5. Multi Device I^2C Reset Configuration Hardware

5.2.5.2 Loading Multiplexed RCW from External Pins (RCW_SRC[0–2] = 000)

When the MSC8156E device is configured to use the multiplexed loading method, it latches all bits of the reset configuration word from the external pins. In this case, the sampled RCW bits are transferred with $\overline{\text{RCW\_LSEL[0–3]}}$ glue logic using the hardware shown in **Figure 5-6**. In this mode, the 64 bits of the RCW are loaded in four passes using only RC[15–0]. RC16 is not used, and RC[20–17] are redefined as the lane select signals ($\overline{\text{RCW\_LSEL[0–3]}}$, respectively), which provide the gating signals for each set of 16 bits transferred. .

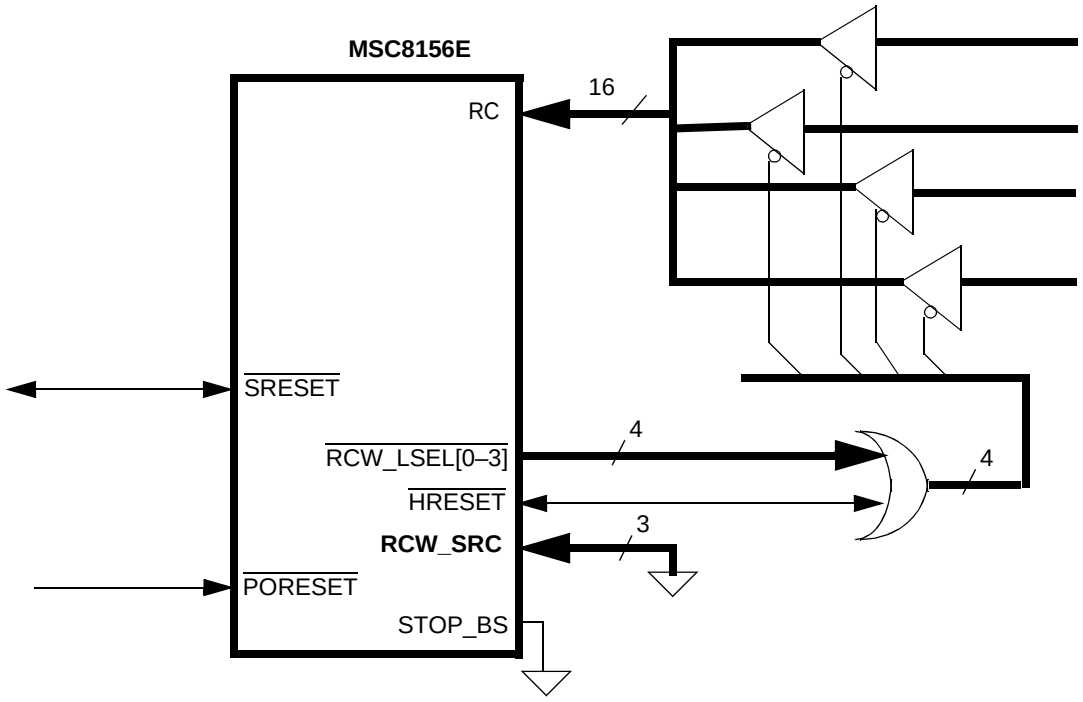


Figure 5-6. Multiplexed External Pins Reset Configuration

Figure 5-2 on page 5-6 shows the timing of the gating signals and indicates which RCW bits are loaded by each lane signal. See **Figure 3-5** on page 3-6 for a detailed description of the lane select signals. The gating summary is as follows:

- Lane 0 ($\overline{\text{RCW\_LSEL0}}$) gates RC[15–0] to RCWLR bits 15–0
- Lane 1 ($\overline{\text{RCW\_LSEL1}}$) gates RC[15–0] to RCWLR bits 31–16
- Lane 2 ($\overline{\text{RCW\_LSEL2}}$) gates RC[15–0] to RCWHR bits 15–0
- Lane 3 ($\overline{\text{RCW\_LSEL3}}$) gates RC[15–0] to RCWHR bits 31–16

5.2.5.3 Loading Reduced RCW From External Pins (RCW_SRC[0–2] = 011)

When the MSC8156E device is configured to use the reduced RCW, the MSC8156E latches some bits of the reset configuration word from external pins. The other bits of the RCWs are loaded from default hard coded values. The hardware connection is shown in **Figure 5-7**.

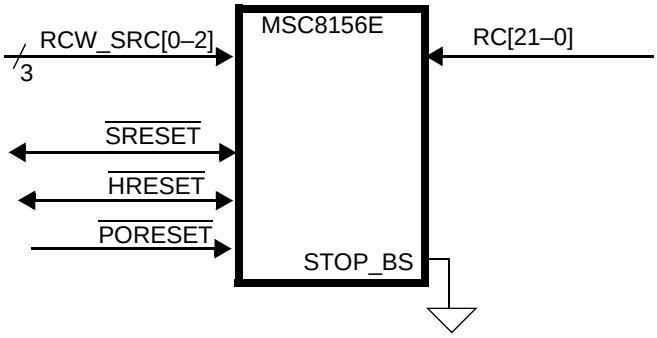


Figure 5-7. External Pins Reduced Reset Configuration

5.2.5.3.1 Reduced External Reset Configuration Word Low Field Values

Table 5-5 defines the combined External and Hard Coded Reset Configuration Word Low field values.

Table 5-5. Combined External and Hard Coded Reset Configuration Word Low Values

Bits	Name	Value	Meaning
31–30	CLKO	00	Select PLL0 divided output clock to be driven on CLKO
29	—	0	Reserved. Should be cleared.
28–24	S2P	0, RC[21–18]	S2P. See **Table 5-9** for details.
22–20	S1P	0, RC[17–15]	S1P. See **Table 5-9** for details.
19–18	—	00	Reserved. Should be cleared.
17	SCLK2	RC14	SerDes2 (RapidIO interface 2) reference clock. A 0 selects 100 Mhz and a 1 selects 125 MHz. See **Table 5-9** for details and limitations.
16	SCLK1	RC14	SerDes1 (RapidIO interface 1) reference clock. This clock is identical to SCLK2 in reduced mode and is driven by RC14.
15–6	—	0000000000	Reserved. Should be cleared.
5–0	MODCK	00, RC[13–10]	MODCK[5–4] = 00, MODCK[3–0] = RC[13–10].

5.2.5.3.2 Reduced External Reset Configuration Word High Field Values

Table 5-6 defines the combined External and Hard Coded Reset Configuration Word High field values.

Table 5-6. Combined External and Hard Coded Reset Configuration Word High Field Values

Bits	Name	Value	Meaning
31–30	—	00	Reserved. Should be cleared.
29	EWDT	0	Disable Watch Dog Timers.
28	PRDY	0	PCI Express not ready.
27–24	BPRT	0, RC[9–7]	See for details **Table 5-10**.
23	RIO	1	RapidIO access enabled.
22	RPT	RC6	RapidIO pass-through enable bit.
21	RHE	0	RapidIO host mode disabled.
20–19	—	00	Reserved. Should be cleared.
18	RM	0	Not reset master.
17	BP	0	Boot patch disabled.
16–13	—	00000	Reserved. Should be cleared.
12	GE1	RC5	0 - TDM2 and TDM3 are driven on pins. 1 - RGMII of GE1 is driven on pins.
11	GE2	RC4	0 - TDM0 and TDM1 are driven on pins. 1 - RGMII of GE2 is driven on pins.
10	R1A	0	SerDes Port 1 RapidIO interface does not accept all.
9	R2A	0	SerDes Port 2 RapidIO interface does not accept all.
8–3	DEVID	00, RC[3–0]	DEVID[5–4] = 00, DEVID[3–0] = RC[3–0]
2	—	0	Reserved. Should be cleared.
1	RMU	0	RMU access local memory on internal port number 0
0	CTLS	1	Common transport type is Large System.

5.2.5.4 Default Reset Configuration Words (RCW_SRC[0–2] = 100 or 101)

When the MSC8156E device is configured not to load the RCW from I^2C EEPROM or external pins, it can be initialized using one of the two hard-coded default options listed in **Table 5-7** and **Table 5-8**.

5.2.5.4.1 Hard Coded Reset Configuration Word Low Field Values

Table 5-7 defines the Hard Coded Reset Configuration Word Low field values.

Table 5-7. Hard Coded Reset Configuration Word Low Field Values

Bits	Name	Value	Meaning
31–30	CLKO	00	Select PLL0 divided output clock to be driven on CLKO
29	—	0	Reserved. Should be cleared.

Table 5-7. Hard Coded Reset Configuration Word Low Field Values (Continued)

Bits	Name	Value	Meaning
28–24	S2P	01010	SerDes Port 2 PCI Express 1x, SGMII1, SGMII2
23–20	S1P	0011	SerDes Port 1 Serial RapidIO 4x 3.125 GHz
19–18	—	00	Reserved. Should be cleared.
17	SCLK2	1	SerDes2 (RapidIO interface 2) reference clock is 125 MHz
16	SCLK1	1	SerDes1 (RapidIO interface 1) reference clock is 125 MHz
15–6	—	0000000000	Reserved. Should be cleared.
5–0	MODCK	000000 000001	RCW_SRC = 100. RCW_SRC = 101

5.2.5.4.2 Hard Coded Reset Configuration Word High Field Values

Table 5-8 defines the Hard Coded Reset Configuration Word High field values.

Table 5-8. Hard Coded Reset Configuration Word High Field Values

Bits	Name	Value	Meaning
31–30	—	00	Reserved. Should be cleared
29	EWDT	0	Disable Watch Dog Timers
28	PRDY	0	PCI Express not ready.
27–24	BPRT	0000	In Hard Coded RCW, this field is ignored, and the boot port is Serial RapidIO interface 1
23	RIO	1	RapidIO access is enabled.
22	RPT	0	RapidIO pass-through is disabled.
21	RHE	0	RapidIO host mode is disabled.
20-19	—	00	Reserved. Should be cleared
18	RM	0	Not reset master
17	BP	0	Boot patch disabled.
16–13	—	00000	Reserved. Should be cleared
12	GE1	1	RGMII GE1 on GPIO pins
11	GE2	1	RGMII GE2 on GPIO pins
10	R1A	0	SerDes Port 1 RapidIO interface does not accept all
9	R2A	0	SerDes Port 2 RapidIO interface does not accept all
8-3	DEVID	000000	Device ID
2	—	0	Reserved. Should be cleared
1	RMU	0	RMU access local memory on internal port number 0
0	CTLS	1	Common transport type is Large System.

5.3 Reset Programming Model

This section describes the following reset registers in detail:

- Reset Configuration Word Low Register (RCWLR), **page 5-17**.
- Reset Configuration Word High Register (RCWHR), **page 5-19**.
- Reset Status Register (RSR), **page 5-22**.
- Reset Protection Register (RPR), **page 5-24**.
- Reset Control Register (RCR), **page 5-25**.
- Reset Control Enable Register (RCER), **page 5-26**.

Note: The Reset register base address is 0xFFF24800.

5.3.1 Reset Configuration Word Low Register (RCWLR)

RCWLR Reset Configuration Word Low Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLKO		—	S2P					S1P				—		SCLK2	SCLK1
Type	R															
Reset	Value depends on the reset configuration word low loaded during reset flow.															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—								PLL1 DIS	—	MODCK					
Type	R															
Reset	Value depends on the reset configuration word low loaded during reset flow.															

The RCWLR is a read-only register set according to the reset configuration word low loaded during the reset flow. **Table 5-9** defines the RCWLR bit fields.

Table 5-9. RCWLR Bit Descriptions

Name	Reset	Description	Settings
CLKO 31–30	0	**CLKOUT Source** This field selects the source for CLKOUT. See **Chapter 7**, *Clocks* for source clock definitions.	00 PLL0 divided output clock. 01 PLL1 divided output clock. 10 PLL2 divided output clock. 11 CLKOUT is always low.
— 29	0	Reserved. Write to zero for future compatibility.	

Table 5-9. RCWLR Bit Descriptions (Continued)

Name	Reset	Description	Settings
S2P 28–24	0	**SerDes2 Protocol** Selects the SerDes protocol to use. **Note:** It is valid to program RCWLR[S1P] and RCWLR[S2P] to have SGMII1 on both ports, SGMII2 on both ports, or to have SGMII1 and SGMII2 on both ports. An internal multiplexer always routes only two physical connections to the QUICC Engine controllers. If the SGMII interfaces are configured by S1P and S2P, SGMII1 (if selected by S1P and S2P) is physically connected to SerDes Port 1 and SGMII2 (if selected by S1P and S2P) is physically connected to SerDes Port 2	00000 No protocol active. 00001 RapidIO 4x 1.25 GHz 00010 RapidIO 4x 2.5 GHz 00011 RapidIO 4x 3.125 GHz 00100 RapidIO 1x 3.125 GHz 00101 RapidIO 1x 1.25 GHz/SGMII1/SGMII2 00110 RapidIO 1x 2.5 GHz/SGMII1/SGMII2 00111 RapidIO 1x 1.25 GHz/SGMII2 01000 Reserved 01001 Reserved 01010 PCI Express 1x/SGMII1/SGMII2 01011 PCI Express 1x/SGMII2 01100 PCI Express 1x/RapidIO 1x 1.25 GHz 01101 PCI Express 4x 01110 PCI Express 2x/SGMII1/SGMII2 01111 Reserved. 10000 PCI Express 1x/RapidIO 1x 2.5 GHz 10001 PCI Express 1x 10010 RapidIO 1x 1.25 GHz 10011 RapidIO 1x 2.5 GHz 10100 RapidIO 1x 2.5 GHz/SGMII2 10101 Reserved. 10110 PCI Express 2x/SGMII2 10111 PCI Express 2x/RapidIO 1x 1.25 GHz 11000 PCI Express 2x/RapidIO 1x 2.5 GHz 11001 PCI Express 2x 11010–11111 Reserved
S1P 23–20	0	**SerDes1 Protocol** Selects the SerDes protocol to use. **Note:** It is valid to program RCWLR[S1P] and RCWLR[S2P] to have SGMII1 on both ports, SGMII2 on both ports, or to have SGMII1 and SGMII2 on both ports. An internal multiplexer always routes only two physical connections to the QUICC Engine controllers. If the SGMII interfaces are configured by S1P and S2P, SGMII1 (if selected by S1P and S2P) is physically connected to SerDes Port 1 and SGMII2 (if selected by S1P and S2P) is physically connected to SerDes Port 2	0000 No protocol active. 0001 RapidIO 4x 1.25 GHz 0010 RapidIO 4x 2.5 GHz 0011 RapidIO 4x 3.125 GHz 0100 RapidIO 1x 3.125 GHz 0101 RapidIO 1x 1.25 GHz/SGMII1/SGMII2 0110 RapidIO 1x 2.5 GHz/SGMII1/SGMII2 0111 RapidIO 1x 1.25 GHz/SGMII1 1000 RapidIO 1x 2.5 GHz/SGMII1 1001 RapidIO 1x 1.25 GHz 1010 RapidIO 1x 2.5 GHz 1011–1111 Reserved.
— 19–18	0	Reserved. Write to zero for future compatibility.	
SCLK2 17	0	**SerDes2 Reference Clock** Selects the SerDes2 reference clock. 100 MHz clock can work for all protocols and frequencies except for 3.125 Gbaud RapidIO; 125 MHz works for all protocols and frequencies with no exceptions.	0 SerDes reference clock = 100 MHz. 1 SerDes reference clock = 125 MHz.

Table 5-9. RCWLR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SCLK1 16	0	**SerDes1 Reference Clock** Selects the SerDes1 reference clock. 100 MHz clock can work for all protocols and frequencies except for 3.125Gbaud RapidIO; 125 MHz works for all protocols and frequencies with no exceptions.	0 SerDes reference clock = 100 MHz. 1 SerDes reference clock = 125 MHz.
— 15–8	0	Reserved. Write to zero for future compatibility.	
PLL1DIS 7	0	**Disable PLL1** Setting this bit disables PLL1. When using clock modes 1 or 37, set this bit to reduce power consumption.	0 PLL1 enabled. 1 PLL1 disabled.
— 6	0	Reserved. Write to zero for future compatibility.	
MODCK 5–0	0	**Clock Mode** Defines the clock operating mode.	See **Chapter 7**, *Clocks*.

5.3.2 Reset Configuration Word High Register (RCWHR)

RCWHR Reset Configuration Word High Register Offset 0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	RC	EWDT	PRDY	BPRT				RIO	RPT	RHE	SBETH	—	RM	BP	—
Type	R															
Reset	Value depends on the reset configuration word high loaded during reset flow.															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—			GE1	GE2	R1A	R2A	DEVID						—	RMU	CTLS
Type	R															
Reset	Value depends on the reset configuration word high loaded during reset flow.															

The RCWHR is a read-only register that derives its values from the reset configuration word high loaded during the reset flow. **Table 5-10** defines the RCWHR bit fields.

Table 5-10. RCWHR Bit Descriptions

Name	Description	Settings
— 31	Reserved. Write to one for future compatibility.	
RC 30	**Selects PCI Express RC Mode** When set, selects the PCI Express root complex (RC) mode of operation. An RC device connects the core processor/memory subsystem to I/O devices while an EP device typically denotes a peripheral or I/O device. See **Section 17.1** for details. This field is always 0 in the preconfigured modes (reduced RCW or hard-coded).	0 PCI Express end point (EP) mode on the PCI Express bus. 1 PCI Express root complex (RC) mode on the PCI Express bus.

Table 5-10. RCWHR Bit Descriptions (Continued)

Name	Description	Settings
EWDT 29	**Enable Watchdog Timers** Selects the status of the software watchdog timers when coming out of reset. The user can override this value by writing to any of the System Watchdog Control Registers (SWCRR[SWEN]) during system initialization.	0 Watchdog timers initially disabled. 1 Watchdog timers initially enabled.
PRDY 28	**PCI Express Ready** Indicates whether the PCI Express controller is ready to be configured.	0 PCI Express not ready. 1 PCI Express ready.
BPRT 27–24	**Boot Port Select** Defines the boot port interface configuration. If RapidIO port is selected as boot port, the RapidIO interface, used for boot port, must be configured to a valid RapidIO protocol using fields S1P/S2P in RCWLR.	0000 I^2C. 0001 RapidIO interface without I^2C 0010 RapidIO interface with I^2C 0011 SPI 0100 RGMII1 without I^2C 0101 SGMII1 without I^2C 0110 RGMII1 with I^2C 0111 SGMII1 with I^2C 1000 RGMII2 without I^2C 1001 SGMII2 without I^2C 1010 RGMII2 with I^2C 1011 SGMII2 with I^2C 1100–1111 Reserved.
RIO 23	**RapidIO Host Access Enable** Enables RapidIO host access to internal memory after boot. When this bit is set, host access is enabled for the following configurations: • BPRT. Chosen host is SGMII. • S1P/S2P. One port is configured as SGMII for boot and the other SerDes does not contain SGMII. **Note:** For both modes, any lane not used for SGMII closes after boot is disabled. To use these lanes after boot, the user should open the lanes as part of the boot code execution.	0 Host access after boot disabled. 1 Host access after boot enabled.
RPT 22	**RapidIO Pass-Through Enable** Selects the reset value of P0PTAACR[PTE] and P1PTAACR[PTE] which determines whether pass-through is disabled or enabled.	0 Pass-through disabled. (P0PTAACR[PTE] and P1PTAACR[PTE] reset value is 0) 1 Pass-through enabled. (P0PTAACR[PTE] and P1PTAACR[PTE] reset value is 1)
RHE 21	**RapidIO Host Enable** Selects whether the RapidIO controller can act as a host. When enabled as host, it uses the base device ID (RapidIO register BDIDCSR) taken from the three least significant bits of the device ID (RCWHR[DEVID]).	0 RapidIO controller is agent. 1 RapidIO controller is host.
SBETH 20	**Simple Boot Over Ethernet** Indicates whether the device uses a simple boot over Ethernet. See **Section 6.2.3** for details. This field is always 0 in the preconfigured modes (reduced RCW or hard-coded).	0 Not simple boot over Ethernet. 1 Simple boot over Ethernet.
— 19	Reserved. Write to zero for future compatibility.	

Table 5-10. RCWHR Bit Descriptions (Continued)

Name	Description	Settings
RM 18	**Reset Master** This bit should be set when the device is a reset master or when booting from a dedicated I^2C EEPROM device.	0 Reset slave. 1 Reset master.
BP 17	**Boot Patch** This bit enables loading patch code for booting from I^2C.	0 Boot patch disabled. 1 Boot patch enabled.
— 16–13	Reserved. Write to zero for future compatibility.	
GE1 12	**GE1 Select** Selects the pins multiplexing between GE1 and TDM[2–3].	0 TDM[2–3] signals are driven on pins. 1 RGMII1 signals are driven on pins.
GE2 11	**GE2 Select** Selects the pins multiplexing between GE2 and TDM[0–1].	0 TDM[0–1] signals are driven on pins. 1 RGMII2 signals are driven on pins.
R1A 10	**RapidIO Interface 1 Accept All** Selects the reset value of P0PTAACR[AA]. When set, RapidIO Interface 1 accepts all device IDs.	0 Do not accept all device IDs. (P0PTAACR[AA] reset value is 0). 1 Accept all device IDs. (P0PTAACR[AA] reset value is 1).
R2A 9	**RapidIO Interface 2 Accept All** Selects the reset value of P1PTAACR[AA]. When set, RapidIO Interface 2 accepts all device IDs.	0 Do not accept all device IDs. (P1PTAACR[AA] reset value is 0). 1 Accept all device IDs. (P1PTAACR[AA] reset value is 1).
DEVID 8–3	**Device ID** Stores the value of the signals sampled during reset.	000000 Master device/Device 0. 000001– 111111 Slave device number (from 1 to 63).
— 2	Reserved. Write to zero for future compatibility.	
RMU 1	**RapidIO RMU Local Memory Access Internal Port Select** Selects the internal port, the RMU uses, to access local memory.	0 Access local memory using internal port 0 [OCN to MBus Bridge 0 (O2M0)]. 1 Access local memory using internal port 1[OCN to MBus Bridge 1 (O2M1)].
CTLS 0	**RapidIO Common Transport Large System** This value is written to the RapidIO register PEFCAR[CTLS]	0 Common transport type is small system 1 Common transport type is large system

Note: The value of fields in the reset configuration word registers (RCWLR and RCWHR) reflect only their state during the reset flow. Some of these parameters and modes can be modified by changing their values in other unit memory mapped registers. Modifying values in other unit memory mapped registers does not affect RCWLR and RCWHR.

5.3.3 Reset Status Register (RSR)

RSR Reset Status Register Offset 0x10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RCWSRC			—					SW0	SW1	SW2	SW3	SW4	SW5	SW6	SW7
Type	R/W															
Reset	RCW_SRC[0–2]			1	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
	—	BSF	SWSR	SWHR	RM	JPO	JH	JS	—				RIO2	RIO1	SRS	HRS
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The reset status register captures various reset events in the device. This register fields are sticky and can be cleared by writing 1, writing 0 has no effect (except RM field which is read-only). **Table 5-11** defines the RSR bit fields.

Table 5-11. RSR Bit Descriptions

Name	Reset	Description	Settings
RCWSRC 31–29	0	**Reset Configuration Word Source** Stores the value of the RCW_SRC[0–2] signals sampled during reset. See **Section 5.2.2**, *Reset Configuration Words Source*. Changing this field has no effect.	000 Multiplexed external RCW loading. 001 I^2C EEPROM in 8-bit addressing mode. 010 I^2C EEPROM in 16-bit addressing mode. 011 Input pins and default settings. 100 Hard coded option 1. 101 Hard coded option 2.
— 28–24	10000	Reserved. Write to 0b10000 for future compatibility.	
SW0 23	0	**Software Watchdog Timer 0** Indicates whether watchdog timer 0 expired.	0 Software watchdog timer 0 not expired. 1 Software watchdog timer 0 expired.
SW1 22	0	**Software Watchdog Timer 1** Indicates whether watchdog timer 1 expired.	0 Software watchdog timer 1 not expired. 1 Software watchdog timer 1 expired.
SW2 21	0	**Software Watchdog Timer 2** Indicates whether watchdog timer 2 expired.	0 Software watchdog timer 2 not expired. 1 Software watchdog timer 2 expired.
SW3 20	0	**Software Watchdog Timer 3** Indicates whether watchdog timer 3 expired.	0 Software watchdog timer 3 not expired. 1 Software watchdog timer 3 expired.
SW4 19	0	**Software Watchdog Timer 4** Indicates whether watchdog timer 4 expired.	0 Software watchdog timer 4 not expired. 1 Software watchdog timer 4 expired.
SW5 18	0	**Software Watchdog Timer 5** Indicates whether watchdog timer 5 expired.	0 Software watchdog timer 5 not expired. 1 Software watchdog timer 5 expired.
SW6 17	0	**Software Watchdog Timer 6** Indicates whether watchdog timer 6 expired.	0 Software watchdog timer 6 not expired. 1 Software watchdog timer 6 expired.
SW7 16	0	**Software Watchdog Timer 7** Indicates whether watchdog timer 7 expired.	0 Software watchdog timer 7 not expired. 1 Software watchdog timer 7 expired.
— 15	0	Reserved. Write to zero for future compatibility.	

Table 5-11. RSR Bit Descriptions (Continued)

Name	Reset	Description	Settings
BSF 14	0	**Boot Sequencer Fail** If set, indicates the I^2C boot sequencer has failed while loading the reset configuration words.	0 No boot sequencer failure. 1 Boot sequencer failure.
SWSR 13	0	**Software Soft Reset** Indicates whether a software soft reset has occurred.	0 No software soft reset. 1 Software soft reset initiated.
SWHR 12	0	**Software Hard Reset** Indicates whether a software hard reset has occurred.	0 No software hard reset. 1 Software hard reset initiated.
RM 11	0	**Reset Master** Indicates whether the device is the reset master.	0 Reset slave. 1 Reset master.
JPO 10	0	**JTAG Power-On Reset** Indicates whether a power-on reset request was received via a JTAG command. When this bit is set, out of reset, it also sets RSR[HRS] and RSR[SRS].	0 No JTAG power-on reset request. 1 JTAG power-on reset request received.
JH 9	0	**JTAG Hard Reset** Indicates whether JTAG hard reset request was received via a JTAG command.	0 No JTAG hard reset. 1 JTAG hard reset initiated.
JS 8	0	**JTAG Soft Reset** Indicates whether JTAG soft reset request was received via a JTAG command.	0 No JTAG reset. 1 JTAG soft reset initiated.
— 7–4	0	Reserved. Write to zero for future compatibility.	
RIO2 3	0	**RapidIO Interface 2 Reset Status** Indicates whether the RapidIO interface 2 received a reset request.	0 No reset request received. 1 Reset request received.
RIO1 2	0	**RapidIO Interface 1 Reset Status** Indicates whether the RapidIO interface 1 received a reset request.	0 No reset request received. 1 Reset request received.
SRS 1	0	**Soft Reset Status** When an external or internal soft reset event is detected, SRS is set.	0 No soft reset event. 1 A soft reset event was detected.
HRS 0	0	**Hard Reset Status** When an external or internal hard reset event is detected, HRS is set.	0 No hard reset event. 1 A hard reset event was detected.

Note: The RSR accumulates reset events. For example, because a software watchdog timer expiration results in a hard reset that in turn results in a soft reset, RSR[SWSR], RSR[SRS], and RSR[HRS] are all set after a software watchdog reset. These must be cleared individually. RSR only returns to its complete reset value when a power-on reset occurs.

5.3.4 Reset Protection Register (RPR)

RPR Reset Protection Register Offset 0x18

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

The RPR enables or disables writing to the reset control register (RCR). The RPR protects unintended software reset requests due to writes to the reset control register (RCR). To enable the RPR, write the value 0x52535445 (“RSTE” in ASCII) to the RPR. When enabled, the control register enable bit in the reset control enable register (RCER[CRE]) is set. Reading the RPR always returns all zeros. To disable writes to the RCR, write a 1 to the RCER[CRE] bit. **Table 5-12** defines the bit fields of RPR.

Table 5-12. RPR Bit Descriptions

Name	Reset	Description
RCPW 31–0	0	**Reset Control Protection Word** Protects unintended software reset request caused by writes to the RCR. Write the value 0x52535445 (“RSTE” in ASCII) to the RCPW to enable the RCR. When the RCR is enabled, the RCER[CRE] bit is set. Reading the RPR always returns all zeros. To disable write to the RCR, write a 1 to RCER[CRE].

5.3.5 Reset Control Register (RCR)

RCR Reset Control Register Offset 0x1C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—													SRH	SWHR	SWSR
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The RCR is used by software to initiate a soft or hard reset sequence. To allow writing to this register, the user must first enable it by writing the value 0x52535445 to the reset protection register (RPR). **Table 5-13** defines the RCR bit fields.

Table 5-13. RCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
SHR 2	0	**Soft Hard Reset** Setting this bit cause the MSC8156E to convert all hard reset flows to soft reset flows. This feature can be used for debug. This bit returns to its reset state during the reset sequence, so reading it always returns a 0.	0 Normal hard reset flow. 1 Hard reset flow converted to soft reset flow.
SWHR 1	0	**Software Hard Reset** Setting this bit cause the MSC8156E to begin a hard reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns a 0.	0 Normal operation. 1 Initiates a hard reset.
SWSR 0	0	**Software Soft Reset** Setting this bit cause the MSC8156E to begin a soft reset flow. This bit returns to its reset state during the reset sequence, so reading it always returns a 0.	0 Normal operation. 1 Initiates a soft reset.

Note: When issuing a reset command by accessing the register, the device enters reset mode immediately. The host transaction does not terminate normally. Always consider the possible outcome when any host is programmed to issue the software reset command.

5.3.6 Reset Control Enable Register (RCER)

RCER Reset Control Enable Register Offset 0x20

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

The reset control enable register shown in indicates by the CRE field that the reset protection register (RPR) was accessed with a value that enables the reset control register (RCR). **Table 5-14** defines the RCER bit fields.

Table 5-14. RCER Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
CRE 0	0	**Control Register Enabled** Indicates the status of the reset control register (RCR). Writing 1 to this bit disables the RCR and clears this bit. Writing zero has no effect.	0 RCR is disabled. 1 The enable value is written to the reset protection register (RPR) to enable the RCR.

Boot Program 6

The boot program initializes the MSC8156E after it completes a reset sequence. The MSC8156E can boot from an external host through the RapidIO interface or download a user boot program through the I^2C, SPI, or Ethernet ports. The default boot code is located in an internal 96 KB ROM at 0xFEF00000–0xFEF17FFF and is accessible to all cores. For readability, the internal boot code is written in C and is based on the Freescale SmartDSP OS.

When cores finish the reset sequence, they all jump to the ROM starting address (0xFEF00000), and run the boot code. Specific tasks may differ based on the core ID.

Note: Boot data is located in the M3 memory at 0xC0101C00–0xC0107FFF (25 KB). Do not write to this area during boot loading.

When the cores finish the boot sequence, they all jump to a user-defined address.

Special conditions for boot code operation include the following:

- The boot code services the watchdog timer if the EWDT bit in the reset configuration word high register (RCWHR) is set (recommended).
- If the boot process fails, the core goes into a debug state and writes an indication of the root error cause to address 0xC0101C04 in M3 memory (see **Section 6.5**, *Boot Errors*, on page 6-23 for details).
- The boot program does not configure the DDR controller. Therefore, if you want to place data in the DDR memory, you must first configure the DDR controller to support the type of DDR memory in the system. To configure the controller, write the configuration data to the DDR controller memory-mapped registers before writing data to the DDR memory. See **Chapter 12**, *DDR SDRAM Memory Controller* for details.
- In any reset state other than PORESET, the device does not execute the multi-device support for using the reset configuration word (RCW) flow with I^2C as described in **Section 6.1.4**. The rest of the boot flow remains the same as described in this chapter.
- The boot code is run by all cores. Task differ by core ID.

Note: Parity error checking is not supported by the boot code because parity errors are unlikely to occur.

6.1 Functional Description

The boot code is divided into four parts shown in **Figure 6-1**:

- *Private configuration (all cores)*. Includes general configuration of all cores.
- *Shared configuration (core 0)*. Includes general configuration of internal CLASS, I^2C, RapidIO interface, and QUICC Engine subsystem.
- *Patch mode*. Allows loading patch code for boot from I^2C.
- *Boot mode select (core 0)*. This part includes downloading of code from one of the MSC8156E bootable ports as defined by the RCWHR[BPRT] field.
- *Boot completion*. All cores complete the boot operation and jump to a user-specified address.

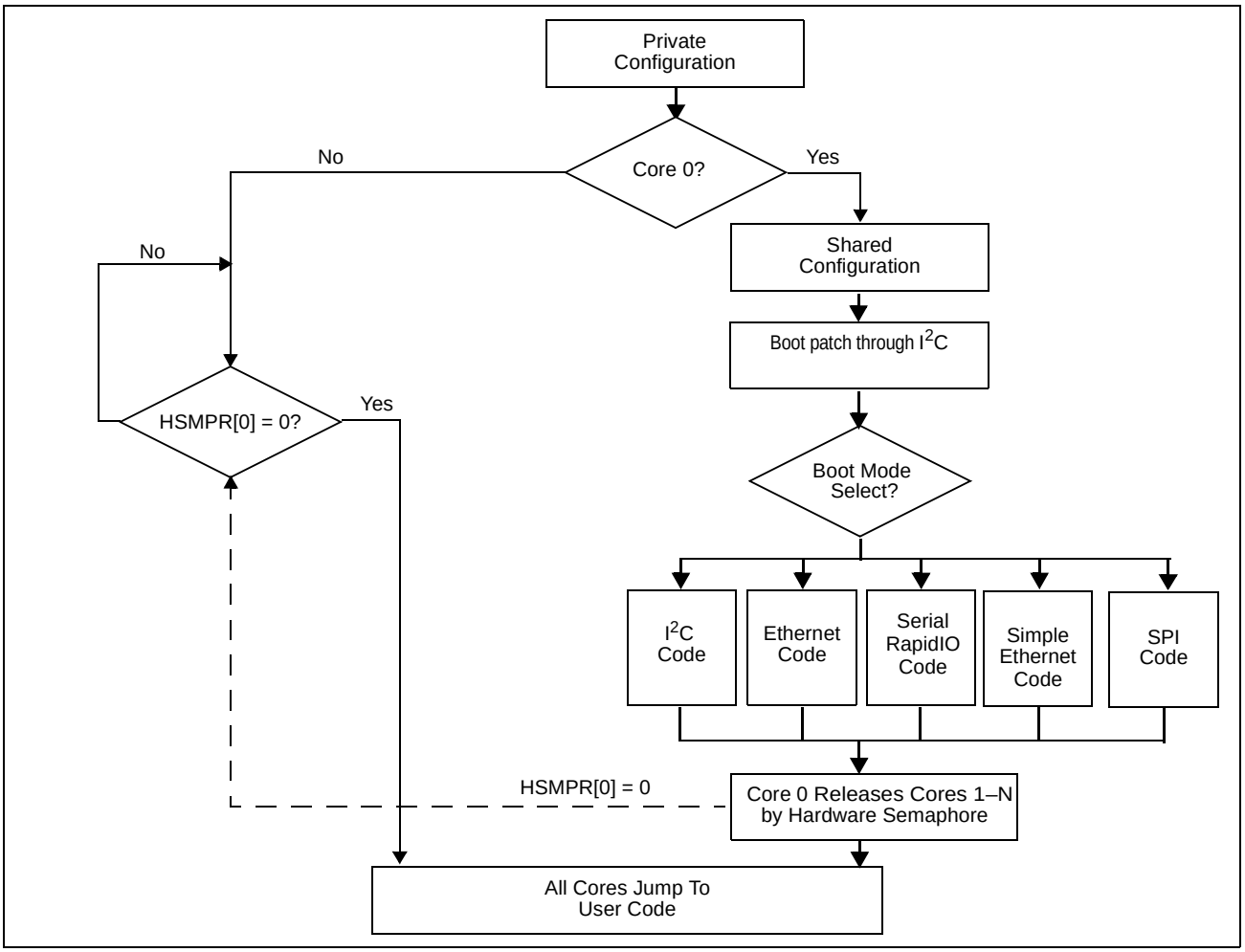


Figure 6-1. Boot Sequence Diagram

6.1.1 Private Configuration

Private configuration includes the following:

- VBA register initialization. The value stored in ROM (0xFEF17000) is used to initialize the Vector Base Address (VBA) register in the SC3850 core. After initialization of the VBA register, any interrupt places the core in debug mode.
- The EPIC is programmed to handle all NMIs correctly.
- The L1 ICache is enabled.
- The stack pointer is set to reside in the M3 memory space dedicated to the boot operation.
- The Error Detection Code (EDC) exception is enabled.

Note: The EPIC, L1 ICache, and MMU are part of the SC3850 DSP core subsystem. See the the *MSC8156 SC3850 DSP Core Subsystem Reference Manual* for configuration details. The manual is only available with a signed non-disclosure agreement (NDA). Contact your Freescale sales office or representative for more information.

6.1.2 Shared Configuration

The shared configuration includes the following:

- If the I^2C controller is used to load the RCW for multi-device slaves (see **Section 6.1.4**, *Multi Device Support for the I2C Bus*, on page 6-4) the necessary number of GPIO lines from the {GPIO[0–3], GPIO[21]} of the master device are set to output and used to drive STOP_BS signals as follows:
  - For up to 5 slaves, the lines drive the signals directly, and the number of lines equals the number of slaves.
  - For up to 15 slaves, using glue logic to drive the STOP_BS signals; the number of required lines is equal to $1 + \lceil \log_2 numSlaves \rceil$.
- If RCWHR[RIO] is set or boot is over the RapidIO interface:
  - LCSBA1CSR is set to 0x1FFE0000, thereby allowing the configuration register space to be physically mapped. This allows configuration and maintenance of the registers through regular read and write operations rather than by maintenance operations.
  - The necessary bits of HSSI_CR1[0:1] are set in order to disable the tri-state on the Serial RapidIO lanes.
  - The necessary bits of P0CCSR and P1CCSR (RapidIO) are set to force 1x or 4x based on RCWLR[S1P] and RCWLR[S2P].
- QUICC Engine module priority is set to be 1 with emergency not masked (that is SDMR[EB1_PR] = 01).

6.1.3 Patch Mode

Patch mode is defined as follows:

- Patch Mode is enabled when RCWHR[BP] is set. See **Chapter 5**, *Reset* for details.
- Boot loads patch code from I^2C and executes in the same manner as boot over I^2C.
- After the patch is executed, the patch code jumps to the address in the boot code defined in address 0xC0101C14. The jump address can be changed by the patch code. By default, the boot code continues Boot Flow from same place after returning from the patch loading.
- Execution continues to load boot code from the boot port defined by RCWHR[BPRT]. If boot port is I^2C, the boot code generates an error and the core goes into a debug state.

6.1.4 Multi Device Support for the I^2C Bus

The MSC8156E can share the I^2C EEPROM device with other MSC8156E devices for loading the reset configuration word (RCW), as well as for reading configuration during boot loading and execution. When the bus is shared, the bus must distinguish among reset masters, reset slaves, and EEPROM slaves:

The reset master (indicated by RCWHR[RM]) holds the $\overline{\text{STOP\_BS}}$ signals of all the slaves high and releases them one by one, thus arbitrating which slave has access to the bus at any moment. When the master deasserts $\overline{\text{STOP\_BS}}$ for a slave, the slave device attempts to access an EEPROM at address 0x57. The actual EEPROM address is 0x50, but the master emulates the EEPROM using address 0x57 to drive the RCW to each slave in turn.

There are a number of assumptions and limitations imposed when multiple devices share the I^2C bus:

1. For each EEPROM in the system, there must be at least one EEPROM master. The EEPROM master is also the reset master (RCWHR[RM])
2. For each EEPROM, there can be 0 or more EEPROM slaves. An EEPROM slave is defined as a device that reads is RCW from the EEPROM and uses data on the bus during boot. The number of EEPROM slaves is stored as a single byte at address 0x8F of the EEPROM.
3. For each EEPROM, there can be 0 or more reset slaves. A reset slave is defined as a device that only reads its RCW from the EEPROM but does not read data from it during boot. The number of reset slaves is stored as a single byte at address 0x11 of the EEPROM.
4. Every EEPROM slave must also be a reset slave.
5. There may be up to 15 reset slaves per EEPROM

6. As a consequence of the conditions listed in 1–5, the limitations on the number of slaves is defined as 0 ≤ #EEPROM slaves ≤ #reset slaves ≤ 15
7. The lowest numbered reset slave must be a higher numbered slave than the highest numbered EEPROM slave (for example, if EEPROM slaves are slaves 0–4, then reset slaves are slaves 5–12).
8. EEPROM slaves must be numbered sequentially from 0 upward.
9. All devices connected to the same EEPROM must have $\overline{\text{PORESET}}$ asserted simultaneously, that is, no single device go through the PORESET sequence without the others.
10. *For multi-device RCW only*. The EEPROM master can have $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted without the slaves being reset as well. Each reset slave can have its $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted without the master being reset.
11. *For multi-device RCW and Boot using I^2C*. If the EEPROM master has its $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted, the EEPROM slaves must have their $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted as well. Each reset slaves can have its $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ asserted without the master being reset. However, there must be external logic that performs the actions that the master performs during its boot sequence. The logic may be implemented as periodic polling by the master, asserting NMI to the reset master, or using an FPGA or other implementation.
12. If there is a shared EEPROM in use in any stage of the reset/boot flow (RCW, serial RapidIO interface configuration, MAC address, I^2C boot), all devices MUST load their RCW from the shared EEPROM.

Note: If the reset master (RCWHR[RM]) fails (due to a stuck I2C_SDA) to read the data at 0x11 or 0x8F of the EEPROM or fails during the sequence of driving the RCW to the reset slaves, the core goes into a debug state and writes the appropriate error code to the M3 memory (see **Section 6.5**).

6.1.5 Example Configuration

Figure 6-2 describes a I^2C multi device system in which MSC8156E #0 is a reset master and MSC8156E #1 is a reset slave. The reset master uses {GPIO[0–3], GPIO[21]} to release the reset slaves. The MSC8156E boot supports up to 15 slaves on a single EEPROM (for RCW). There are two possibilities as to how the reset slave $\overline{\text{STOP\_BS}}$ signals are handled:

- If there are 5 slaves or less, connect each GPIO line directly to one of the slaves. The master deasserts and asserts the lines when necessary.
- If there are more than 5 slaves, GPIO[21] is used to latch the values of GPIO[0–3] into the decoder glue logic (latch when high). This value indicates which of the slave $\overline{\text{STOP\_BS}}$ signals to pull low. When an all 1 values are latched, all $\overline{\text{STOP\_BS}}$ signals should be pulled high.

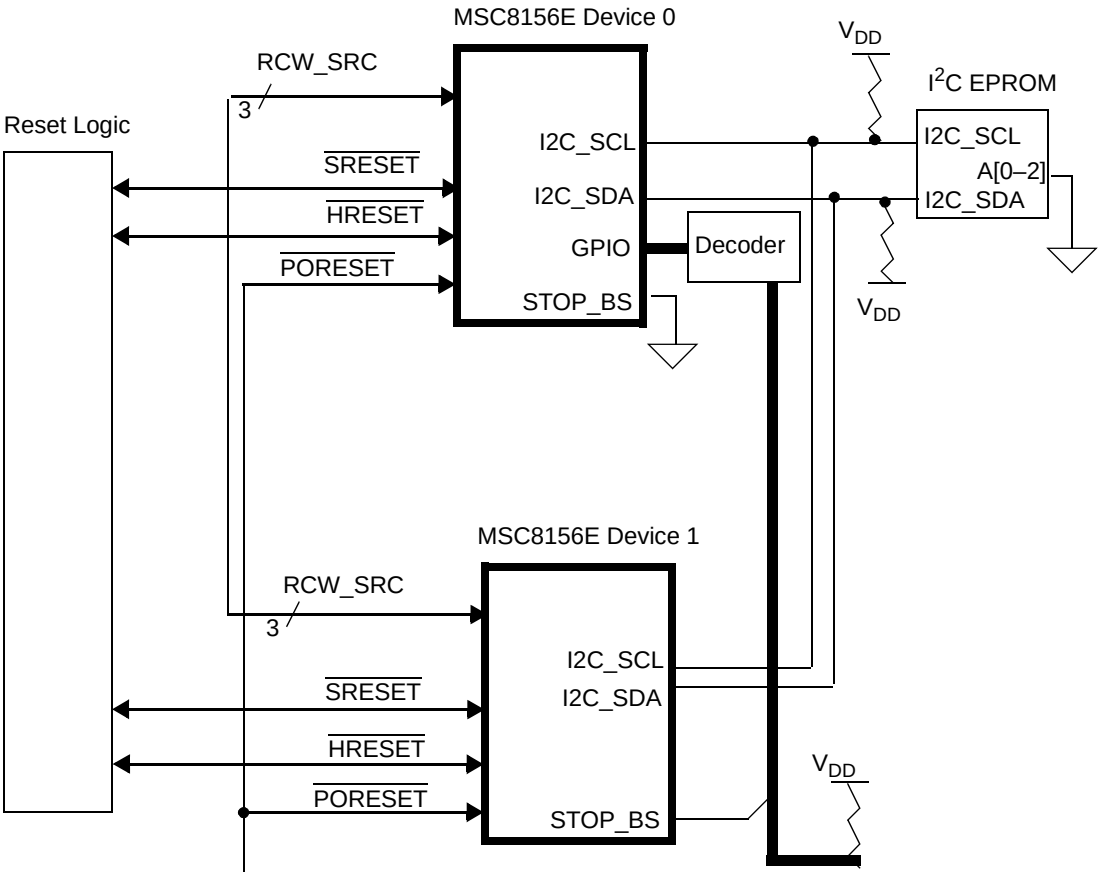


Figure 6-2. I^2C Multi Device System

Figure 6-3 shows the I^{2}C initialization and multi device support.

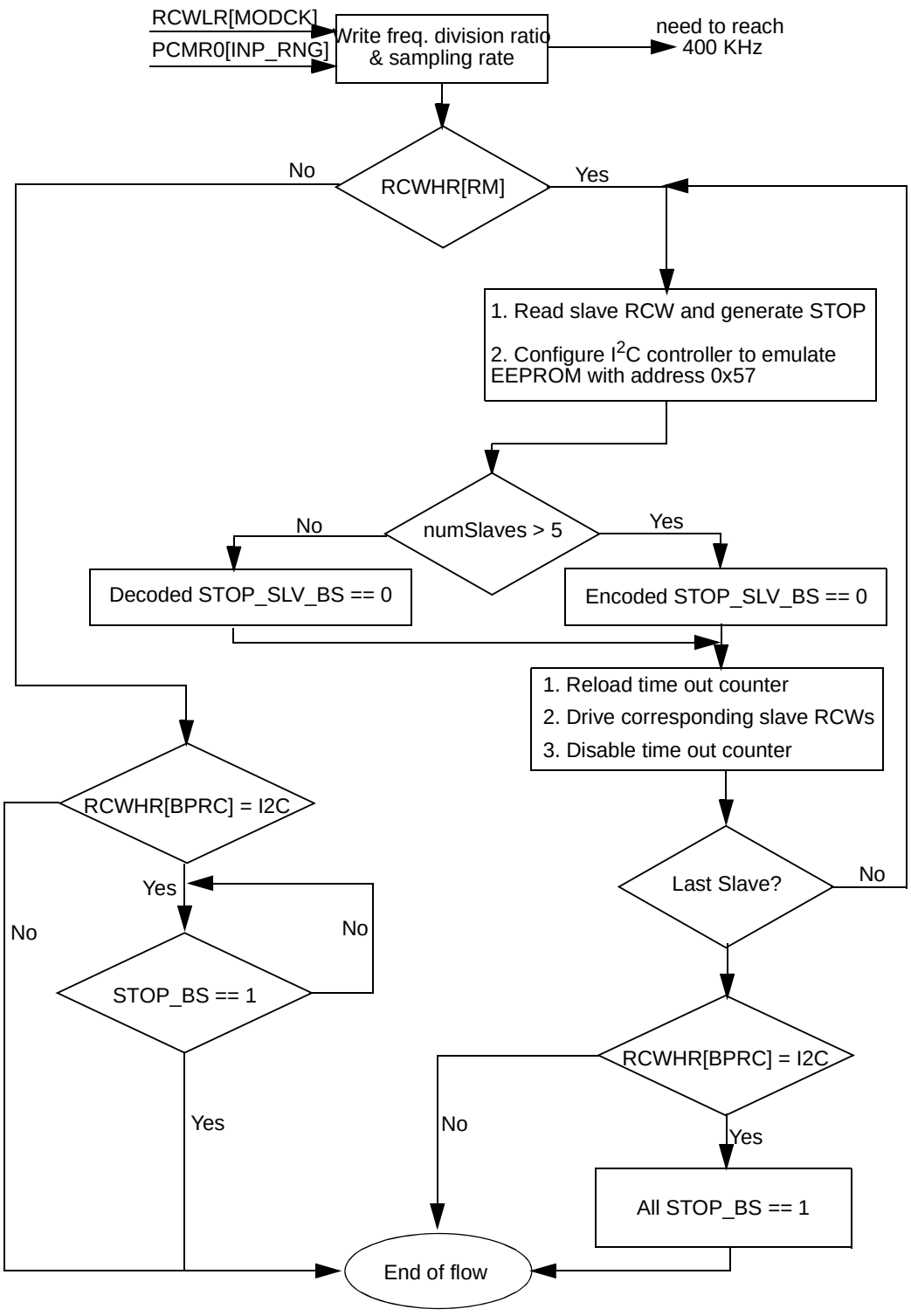


Figure 6-3. I^{2}C initialization and Multi Device Support

The following stages are performed to serve as the master chip on a multi-device board.

1. The MSC8156E reads RSR to determine if the reset is PORESET. If it is not PORESET, this section of the boot is bypassed entirely.
2. The MSC8156E reads RCWHR[RM] to determine if it is the master on the multi-device board or a slave on the multi-device board.
3. If the MSC8156E is the master on the multi-device board:
   a. I2CFDR and I2CDFSSR are programed based on the following data
      — RCWLR[MODCK]
      — RSR[RSTSRC]
      The frequency used for I2C_SCL is set as closely as possible to 400 kHz.
   b. The Reset master reads the slaves RCW from their location in the I^2C EEPROM (with address 0x50) and stores them into M3.
   c. The reset masters I^2C controller is configured to work as an EEPROM (slave mode) with address 0x57.
   d. The reset master deassert $\overline{\text{STOP\_BS}}$ for the current slave (directly or via the decoder).
   e. The reset master:
      • Drives preamble 0xAA55AA
      • Drives header 0xFFFFFF
      • Drives RCWLR
      • Drives header 0xFFFFFF
      • Drives RCWHR
      for each read request of the reset slave, and then generates a STOP condition on the bus in order to free it up because the slave I^2C controller does not generate a STOP condition.
   f. Repeat steps a–e for all slaves.
   g. {GPIO[0–3}, GPIO[21]} are set to 0x1F thus deasserting all the slaves $\overline{\text{STOP\_BS}}$ signals, and the master can continue performing its boot flow.
4. After getting its RCS, the reset slave starts to run the boot program.

Note: If the MSC8156E is a an EEPROM slave, the boot waits until $\overline{\text{STOP\_BS}}$ is to pulled high before continuing with the boot program, thus allowing all reset slaves to read their reset word before any device tries to access the EEPROM for boot code.

6.2 Boot Modes

The Boot Mode is selected by the value in the RCWHR[BPRT] field. The following sections describe the operation of each boot mode.

6.2.1 I²C EEPROM

The MSC8156E boot expects the I²C EEPROM to be divided in to four sections:

1. Reset words. This section starts at address 0x0000 of the EEPROM and includes the reset words for the reset master, an indication as to the number of reset slaves and the reset words for all the slaves.
2. Reserved.
3. Boot configuration. This section starts at address 0x0090 of the EEPROM and can contain one of the following:
   - MAC addresses for up to 64 devices (6 bytes per address). The boot knows to associate each address with the appropriate device based on an offset of 6 × RCWHR[DEVID]. The expected format is consecutive 6 byte fields.
   - Serial RapidIO configuration. This option allows the user to configure up to 47[1] registers and should be used to set the appropriate values of HSSI_CR1 and HSSI_CR2 in the general configuration block. The expected format is address, data pairs. The 8 bytes following the last pair should always be set to {0xFFFFFFFF, 0xFFFFFFFF}, regardless of the actual number of pairs placed in the EEPROM to signal that no more configurations are necessary.
4. Boot code. This section starts at address 0x0210 of the EEPROM and contains the user code. The boot code must be of the size (n × 4) + m[bytes], where n is any integer greater than or equal to 0 and m is either 0 or 1. If n is larger than 0, the value in the Destination Address field must be 32-bit aligned.

Note: Although not all sections are used by all boot options, the section addresses are fixed. However, any section not used by a specific boot option can be used for general purposes within the following guideline examples:

- In a system with only two DSP devices that supports multi-device boot, only addresses 0x0 to 0x19 within the reset word section must carry the appropriate valid values, as shown in **Figure 6-4**. Addresses 0x20 to 0x89 are available for any general-purpose use.
- For boot scenarios without I²C support, addresses 0x90 to 0x20F are available for any general-purpose use.

1.47 pairs along with 8 bytes end flag of {0xFFFFFFFF, 0xFFFFFFFF} is the amount that fits in the same space as 64 MAC addresses ($\lfloor (64 \cdot 6)/(2 \cdot 4) \rfloor = 47 + 1$)

- For boot scenarios that do not use I^2C and do not use boot patch, addresses 0x210 and above are available for general-purpose use.
- Addresses 0x11 (number of reset slaves) and 0x8F (number of EEPROM slaves) must be used for their defined function for all boot options. These addresses are never available for general-purpose use.

Figure 6-4 shows a complete example of an EEPROM contents:

Address	0	1	2	3	4	5	6	7	Description
0x0000	1	0	1	0	1	0	1	0	**Preamble**
0x0001	0	1	0	1	0	1	0	1	
0x0002	1	0	1	0	1	0	1	0	
0x0003	1	1	1	1	1	1	1	1	**Master Reset Configuration Word Low Preload Command**
0x0004	1	1	1	1	1	1	1	1	
0x0005	1	1	1	1	1	1	1	1	
0x0006	Reset Configuration Word Low [31–24]								
0x0007	Reset Configuration Word Low [23–16]								
0x0008	Reset Configuration Word Low [15–8]								
0x0009	Reset Configuration Word Low [7–0]								
0x000A	1	1	1	1	1	1	1	1	**Master Reset Configuration Word High Preload Command**
0x000B	1	1	1	1	1	1	1	1	
0x000C	1	1	1	1	1	1	1	1	
0x000D	Reset Configuration Word High [31–24]								
0x000E	Reset Configuration Word High [23–16]								
0x000F	Reset Configuration Word High [15–8]								
0x0010	Reset Configuration Word High [7–0]								
0x0011	$numResetSlaves \in [0-15]$								**Number of Reset Slaves**
0x0012	Reset Configuration Word Low [31–24]								**Reset Configuration Word Low of Slave 1**
0x0013	Reset Configuration Word Low [23–16]								
0x0014	Reset Configuration Word Low [15–8]								
0x0015	Reset Configuration Word Low [7–0]								
0x0016	Reset Configuration Word High [31–24]								**Reset Configuration Word High of Slave 1**
0x0017	Reset Configuration Word High [23–16]								
0x0018	Reset Configuration Word High [15–8]								
0x0019	Reset Configuration Word High [7–0]								
	.....								
	.....								
0x0082	Reset Configuration Word Low [31–24]								**Reset Configuration Word Low of Slave 15**
0x0083	Reset Configuration Word Low [23–16]								
0x0084	Reset Configuration Word Low [15–8]								
0x0085	Reset Configuration Word Low [7–0]								

Figure 6-4. EEPROM Contents

Address	0	1	2	3	4	5	6	7	Description
0x0086	Reset Configuration Word High [31–24]								Reset Configuration Word High of Slave 15
0x0086	Reset Configuration Word High [23–16]								
0x0088	Reset Configuration Word High [15–8]								
0x0089	Reset Configuration Word High [7–0]								
0x008A	.....								Reserved
0x008B	.....								
0x008C	.....								
0x008D	.....								
0x008E	.....								
0x008F	15 ≤ numResetSlaves ≤ numEEPROMslaves ≤ 0								Number of EEPROM Slaves
0x0090									Configuration/MAC Address First Byte
0x020F									Configuration Last Byte/ Device 0x2F Last MAC Address Byte
0x0210	1	0	1	1	1	1	1	1	Block Control (Block #0)
0x0211	size[0–7]								Block Size
0x0212	size[8–15]								
0x0213	size[16–23]								
0x0214	NBA[31–24]								Next Block Address
0x0215	NBA[23–16]								
0x0216	NBA[15–8]								
0x0217	NBA[7–0]								
0x0218	DA[31–24]								Destination Address
0x0219	DA[23–16]								
0x021A	DA[15–8]								
0x021B	DA[7–0]								
	CODE								Payload Data
	Checksum[15–8]								Checksum and Checksum
	Checksum[7–0]								
	Checksum[15–8]								
	Checksum[7–0]								
	1	0	1	1	1	1	1	1	Block Control (Block #1)
	.....								
End of EEPROM									

Note: The value shown for Block Control is an example only.

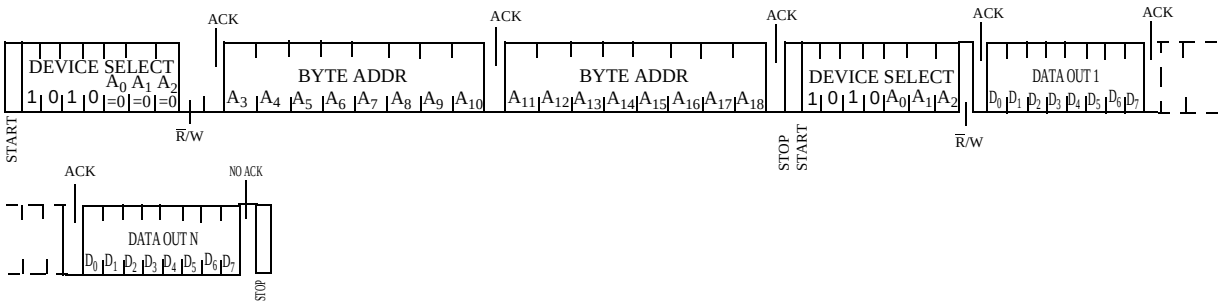
Figure 6-4. EEPROM Contents (Continued)

The I^2C boot loading is performed with the I^2C controller. To allow for EEPROMs of up to 64 Kbytes, 19-bit addressing is used. The 7 most significant bits (msb) of the I^2C slave address are always 0b1010000. The I^2C controller expects a specific memory image when trying to read data from the EEPROM. The I^2C memory image consists of the following:

1. *Block Control***.** A 1-byte control field that contains:
   - 1 bit of CSE. A 1 indicates that the checksum is enabled.
   - 1 reserved bit that should be cleared (0).
   - 6 bits of CHIP_ID indicate the destination chip. 0x3F means broadcast.
2. *Block Size***.** These 3 bytes represent the number of bytes in the payload data field (e.g if the payload size is 12 bytes, Block Size = {0x00, 0x00, 0x0C}).
3. *Next Block Address*. The address in which the next block is located. If the next block address equals 0x0 the bootloader assumes that the next block is sequential. If next block address equals 0xFFFFFFFF, this block is the end block.
4. *Destination Address*. The address to which the payload data should be written.
5. *Payload Data***.** Holds 1≤size<216 bytes (up to 64 Kbytes) of data to be written to on-device memory according to the destination address.
6. *Checksum***.** A 2-byte field that holds the XOR of all previous data (Block Control and on). The boot code XORs each received 2 bytes with the previous checksum value and verifies the validation by comparing it to this field.
7. $\overline{Checksum}$**.** A 2-byte field that holds bitwise-not of the Checksum.

The I^2C bootloader expects the 4 bytes of Checksum and $\overline{\text{Checksum}}$ regardless of the CSE value. If the Checksum is disabled, these 4 bytes are not checked. By using Checksum and $\overline{\text{Checksum}}$, the boot ensures that all values of the bits are real values and that there are no stuck signals. If both Checksum and $\overline{\text{Checksum}}$ are erroneous in a block, core 0 enters the debug state.

The I2C_SCL frequency is set as closely to 400KHz as possible, as mentioned in **Section 6.1.4**. For each block, the Software I^2C read access begins with the boot code driving the device select ({A0,A1,A2} = 0b000) and 2 bytes of address, followed by a RESTART condition. The I^2C slave drives its data (beginning with the Block Control byte) until the end of the block. The last byte of each block is not acknowledged by the MSC8156E. After the ninth unacknowledged bit, the boot code generates a STOP condition. **Figure 6-5** describes the Software I^2C read access.



Note: A_0 and D_0 are the most significant bits.

Figure 6-5. I^2C Read Access

6.2.2 Ethernet

- The MSC8156E device can load files through the Ethernet port using DHCP (Dynamic Host Configuration Protocol) and TFTP (Trivial File Transfer Protocol).Supports RGMII @1000 Mbps and SGMII @1000 Mbps full duplex.
- For DHCP, each client must have its own unique MAC (Media Access Control) address. This MAC address can be based on RCWHR[DEVID] or be user-defined.
- This DHCP implementation supports IPv4.

Booting over Ethernet is enabled on GE1 (UCC1) and GE2 (UCC3) depending on the values of the following bit fields:

- RCWHR[BPRT]. Determines which ports to use for boot loading.
  - 0x4 = RGMII1 without I^2C
  - 0x5 = SGMII1 without I^2C
  - 0x6 = RGMII1 with I^2C
  - 0x7 = SGMII1 with I^2C
  - 0x8 = RGMII2 without I^2C
  - 0x9 = SGMII2 without I^2C
  - 0xA = RGMII2 with I^2C
  - 0xB = SGMII2 with I^2C
- RCWHR[GE1]. Determines whether the signals lines are TDM[2–3] or RGMII1.
- RCWHR[GE2]. Determines whether the signal lines are TDM[0–1] or RGMII2
- RCWLR[S1P]. Determines which SGMII signals are enabled on SerDes port 1:
  - 0x5 = SGMII1 and SGMII2.
  - 0x6 = SGMII1 and SGMII2
  - 0x7 = SGMII1
  - 0x8 = SGMII1
- RCWLR[S2P]. Determines which SGMII signals are enabled on SerDes port 2.
  - 0x5 = SGMII1 and SGMII2
  - 0x6 = SGMII1 and SGMII2

— 0x7 = SGMII2
— 0xA = SGMII1 and SGMII2
— 0xB = SGMII2
— 0xE = SGMII1 and SGMII2
— 0x14 = SGMII2
— 0x16 = SGMII2

Note: It is valid to program RCWLR[S1P] and RCWLR[S2P] to have SGMII1 on both ports, SGMII2 on both ports, or to have SGMII1 and SGMII2 on both ports. An internal multiplexer always routes only two physical connections to the QUICC Engine controllers. If the SGMII interfaces are configured by S1P and S2P, SGMII1 (if selected by S1P and S2P) is physically connected to SerDes Port 1 and SGMII2 (if selected by S1P and S2P) is physically connected to SerDes Port 2

Figure 6-6 describes the Ethernet bootloader flow.

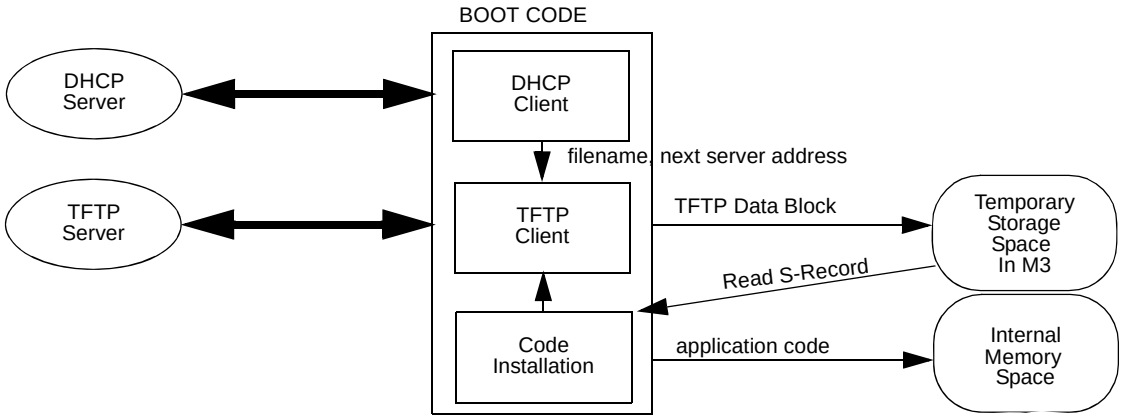


Figure 6-6. Ethernet Bootloader Flow

:The Ethernet bootloader flow includes:

1. Configuring the QUICC Engine drivers based on RCWHR[BPRT].
2. Finding a DHCP server and receive configuration parameters (filename, server address, and so on).
3. Reading a block of the boot file, in S-Record format, from a TFTP server.
4. Processing each TFTP data block and placing it in its memory destination.
5. Sending a TFTP acknowledge to the TFTP server.
6. Repeating steps 2–5 until the end of the data is transferred.

6.2.2.1 DHCP Client

The basic steps that occur when a DHCP client requests an IP address from a DHCP server are shown in **Figure 6-7**.

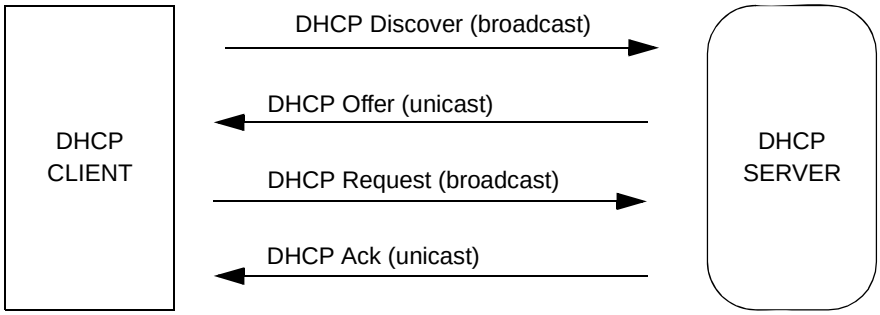


Figure 6-7. DHCP Transactions

- The client sends a DHCP DISCOVER broadcast message to locate a DHCP server.
- A DHCP server offers configuration parameters (such as an IP address, a TFTP server IP address, bootfile name...).
- The client returns a formal request for the first offered IP address to the DHCP server in a DHCPREQUEST broadcast message.
- The DHCP server confirms that the IP address has been allocated to the client by returning a DHCPACK unicast message to the client.

There are two possibilities for setting the MSC8156E MAC address during the boot:

- User defined and read from an I^2C EEPROM. See **Section 6.2.1** for details.
- Predefined default using the following fields:
  - A constant of 32 bits: {0x1E, 0xF7, 0xD5, 0x00}
  - 8 bits consisting of (RCWHR[DEVID]) (aligned to the right and padded with 0)
  - A constant of 8 bits: {0x00}

The predefined option is configured to be an individual locally administered address in accordance with **IEEE** Std. 802-2001™. This MAC address scheme allows for more that 8 unique MAC addresses per device by changing the last 4 bit values, thus allowing each core to have 2 MAC addresses for use during operational mode.

6.2.2.2 TFTP Client

This implementation supports three types of messages: TFTP REQUEST, TFTP DATA and TFTP ACK. **Figure 6-8** describes the TFTP handshake.

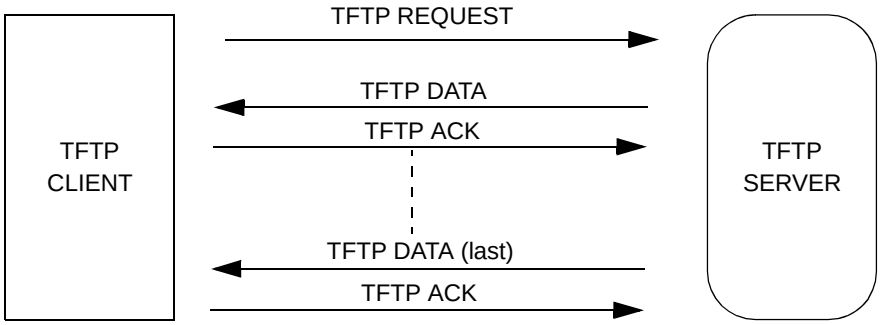


Figure 6-8. TFTP Transactions

- The TFTP transfer is initiated by the client when it issues a TFTP REQUEST (which contains the file name to download).
- In response, the server provides the application with a series of TFTP DATA messages.
- The client handshakes each data block by issuing a TFTP ACK allowing the server to proceed with subsequent TFTP DATA messages.
- This process repeats until all data blocks are received.

6.2.2.3 Boot File Format

The Ethernet bootloader expects an application file in the form of an S-Record file. The S-Record file is a text representation of the binary program code. The S-Record file structure is describes in **Figure 6-9**.

Figure 6-9. S-Record File Structure

Each line of an S-Record file corresponds to any of the following: start record, data record, or end record. Each record is terminated with a line feed.

Note: The S-Record that is downloaded during boot over Ethernet should include no whitespaces (including newlines).

A record has the following format:

S<type><length><address><data><checksum>

Note: This implementation supports only record of types: S0, S3 or S7.

- The description of fields is described in **Table 6-1**.

Table 6-1. S-Record description of Fields

Field	Width in Characters	Description
S<type>	2	The type of record (S0, S3 or S7)
<length>	2	The count of remaining character pairs in the record
<address>	4	The address at which the data field is to be loaded into memory
<data>	≤64	The memory loadable data or descriptive information
<checksum>	2	The least significant byte of the ones complement of the sum of the byte values represented by the pairs of characters making up the length, the address, and the data fields

- *S0 Record***.** Starting record. The address and data fields are ignored and checksum check is executed.
- *S3 Record***.** Data record. The address field is interpreted as a 4-byte address. The data field is composed of memory loadable data.

- *S7 Record***.** Termination record. The address fields is interpreted as the 4-byte address to which to jump after boot. No checksum check is executed.

Shown below is a typical S-record format file:

```
S0030000FC
S30D00002FE731DC3180BEF09E7062
S705000000000C
```

The S0 record is composed as follows:

- **S0.** Indicating it is a starting record.
- **03.** Hexadecimal 03 (decimal 3). Indicating that three character pairs (or ASCII bytes) follow.
- 0000. Information string (ignored)
- **FC.** Checksum field.

The S3 record is composed as follows:

- **S3.** Indicating it is a data record to be loaded at a 4-byte address.
- **0D.** Hexadecimal D (decimal 13), representing a 4 byte address, 8 bytes of binary data, and a 1 byte checksum, follow.
- **00002FE7.** Eight character 4-byte address field.
- **31DC3180BEF09E70.** 8-character pairs representing the actual binary data.
- **62.** Checksum field.

The S7 record is composed as follows:

- **S7.** Indicating it is the last record.
- **05.** Hexadecimal 05 (decimal 5). Indicating that five character pairs (4 byte address and a 1 byte checksum follow).
- 00000000. Address field to jump to at the end of boot (is written to 0xC0101C10).
- **0C**. Checksum field.

Each S-Record line includes an address field that maps the lines data content to a memory location in MSC8156E. Core 0 moves the data to this address.

Note: Because the MSC8156E uses 32-bit addressing, use of S3 and S7 is recommended.

6.2.3 Simple Ethernet Boot

The MSC8156E supports a simple Ethernet boot mode. This mode is selected by setting the RCWHR[SBETH] bit during the $\overline{\text{PORESET}}$ sequence (see **Section 5.3.2** for details).

6.2.3.1 Simple Ethernet Boot Flow

The simple Ethernet boot mode uses the following sequence for processing:

- The bootloader configures the QUICC Engine subsystem drivers based on the RCWHR[BPRT].
- There are two possibilities for setting the MAC address during the boot. See **Section 6.2.2.1** for details.
- When the Ethernet interface is configured, the bootloader sends the start handshake data 0x17171717.
- The boot data is read a block at a time by the bootloader using a simple Ethernet frame format:

  <MAC Dest><MAC Source><Type><2 Bytes Data_Length><4 Bytes Address><Data>
- Simple boot over Ethernet packets should have the ethertype field of the MAC header set to 0x0004.
- The bootloader processes each data block: <data_length><address><data> and places it in the destination memory location.
- The bootloader continues to process blocks until the end handshake data 0xA5A5A5A5 is written to address 0xC0101C00.
- All cores jump to the address written in 0xC0101C10. The value in the address should be written during the boot loading process.

6.2.3.2 Simple Ethernet Boot Ports

Booting over Ethernet is enabled on GE1 (UCC1) and GE2 (UCC3) depending on the values of RCWHR[BPRT], RCWHR[GE1], RCWHR[GE2], RCWLR[S1P] and RCWLR[S2P].

Note: Use the following guidelines to configure bits to support the selected boot modes:

- For boot over Ethernet in RGMII mode the following bits should be configured:
  - RCWHR[GE1] = 1 for boot over port 1
  - RCWHR[GE2] = 1 for boot over port 2

6.2.3.3 Boot File Format

The Ethernet bootloader expects an application file in the format of a Simple Ethernet frame. The Simple Ethernet frame has the following format:

<Length><Address><Data>

The description of fields is described in **Table 6-2**.

Table 6-2. Simple Ethernet Description of Fields

Field	Width in Characters	Description
<length>	2	The count of remaining character pairs in the record
<address>	4	The address at which the data field is to be loaded into memory
<data>		The memory loadable data or descriptive information

Each Simple Ethernet address field maps the data content to a memory location in the DSP. Core 0 moves the data to this address. The following example shows a typical Simple-Ethernet packet for End-of-handshake between the Ethernet master and the MSC8156E boot:

```
1E7FD5000000
1E7FD5100000
0004
0004
C0101C00
A5A5A5A5
```

The End-of-handshake packet sent by the Ethernet Master comprises the following:

- 1E7FD5000000, Destination MAC address (default MSC8156E address)
- 1E7FD5100000. Source MAC address.
- 0004. Ethernet type. The MSC8156E expects Ethernet type 0x0004.
- 0004. Length is 4 bytes.
- C0101C00. The address at which the data field is to be loaded into memory. This is the Handshake address for the MSC8156E.
- A5A5A5A5. Handshake data.

6.2.4 Serial RapidIO Interconnect

In this procedure a Serial RapidIO master waits for the MSC8156E boot program to finish its default initialization and then initializes the device by typically loading code and data to the on device memory.

6.2.4.1 Serial RapidIO Without I^{2}C Support

The boot code configures HSSI_CR[0–1] and PxCCSR according to RCWLR[S1P] and RCWLR[S2P] and writes 0x17171717 to address 0xC0101C00. The flow is:

1. Disable port by writing to PxCCS register.
2. Set x1 or x4 by writing to PxCCSR.
3. Enable lanes by writing to the HSSI_CR1[0–1].
4. Enable the port by writing to PxCCSR.
5. Afterwards, it polls this address until the serial RapidIO master writes 0xA5A5A5A5 to it, thereby indicating that it has finished its code loading.

Figure 6-10 describes the boot flow from Serial RapidIO interconnect**.**

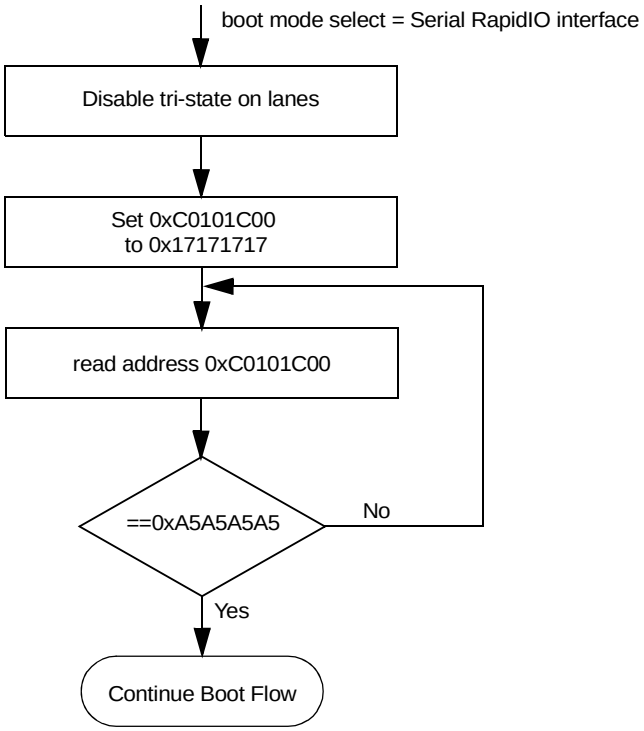


Figure 6-10. Serial RapidIO Interface Boot Flow

6.2.4.2 Serial RapidIO Interface with I²C Support

The user can place {addr, data} pairs in the I²C EEPROM to configure various registers. The address field should be as seen by the SC3850 DSP core. This feature is most commonly used to configure registers in the general configuration block (see **Section 6.2.1**, *I2C EEPROM*, on page 6-9 for details). The boot supports up to 47 such pairs. The 8 bytes following the last pair should always be set to {0xFFFFFFFF, 0xFFFFFFFF}, regardless of the actual number of pairs placed in the I²C EEPROM.

Note: Multiple devices connected to a shared EEPROM see the same address/data pairs.

6.2.5 SPI

The MSC8156E can boot from a Flash memory on the SPI. The boot expects a Flash memory that latches on the rising edge of the clock and on which data is valid after the falling edge. The chip-select should be a $\overline{CS}$ low signal. The boot code expects to see the same data format used for the I²C EEPROM (see described in **Section 6.2.1**, *I2C EEPROM*, on page 6-9, item 4 for details on the boot code requirements) starting at address 0 of the SPI.

A shared SPI bus is arbitrated by all the devices connected to it by polling $\overline{CS}$. All signals should be connected as open-drain if more than one device is connected to the SPI flash. The SPI bus will run no faster than 400 KHz to support multiple devices connected with an open drain.

Note: If the RCW is read from EEPROM, the device for which RCWHR[RM] equals 1 should have RCWHR[DEVID] of 0. Using this configuration setting saves on arbitration cycles towards the SPI flash.

Note: During boot over SPI, the pins described in **Chapter 22**, *GPIO* are used per their SPI functionality. In addition, GPIO23 is used as a chip-select signal ($\overline{CS}$) to control access to the SPI Flash memory.

6.3 Jump to User Code

Before finishing its tasks the boot code preforms these actions:

- If RCWHR[RIO] is cleared, the boot code disables host accesses by RapidIO interface to internal memory space by putting the lanes into tri-state high impedance state.
- Invalidate all range of ICaches and close MMU program windows.
- Core internal registers (other than R0 and VBA) are set to 0x00000000.
- All configurations which were done by the boot code are cleared.
  - Module registers
  - GPIO configurations
  - Write 0x00000000 to GIER (see **Chapter 8**, *General Configuration Registers*) to clear the register.

  - Disable all interrupts (NMI excluded)
  - Clear all QUICC Engine registers by writing 1 to QECMDR[RST]
- 0x900D900D is written to 0xC0101C0C in M3 indicating that the boot has finished executing.
- All cores jump to the address written in 0xC0101C10. The value in this address should be written during the boot loading process.

6.4 System after Boot

- All MATTs in the MMUs are set to their reset value values (except M_xSDBx[PBS] which is set to 0x2).
- L1 I-Cache is enabled, but there are no cacheable windows.
- All NMIs will be configured as NMIs in the EPIC.
- VBA equals 0xFEF17000. Any interrupt in the EPIC puts the core in debug.
- EDC is enabled.
- Core register values are not guaranteed and should be initialized before use.

6.5 Boot Errors

If the boot code fails, an indication as to the root cause is written to 0xC0101C04. The possible reasons are listed in **Table 6-3**.

Table 6-3. Boot Error Codes

Error Code	Description
0x003FEFFF	Catastrophic Error. SmartDSP OS Function failed
0x003FEFFE	DHCP server time out
0x003FEFFD	Corrupted boot file. The possible causes are: • Checksum wrong in TFTP file • Checksum wrong in I^2C file • Unsupported S-Record type
0x003FEFFC	TFTP server time out
0x003FEFFA	TFTP server sent ERROR code (05)
0x003FEFF9	Unsupported boot port
0x003FEFF8	Reset slave does not respond
0x003FEFF5	Boot over Ethernet or RapidIO isn't supported with current RCW configuration
0x003FEFF4	Too many I^2C RCW slaves in address 0x11 of the EEPROM
0x003FEFF3	Error in protocol between I^2C RCW master and slave
0x003FEFF2	Boot port trying to write to area designated for boot (0xC0101C18–0xC0107FFF)
0x003FEFF1	In patch mode, the boot port is I^2C.
0x0027EFFE	Lost arbitration on I^2C us.
0x0027EFFD	Time-out on I^2C acknowledge (9th clock).

Table 6-3. Boot Error Codes

Error Code	Description
0x0027EFFC	Stuck I2C_SDA (I^2C bus).
0x00000000	Unexpected debug condition in the SC3850 Core (unexpected interrupt, EE0 asserted and so on)

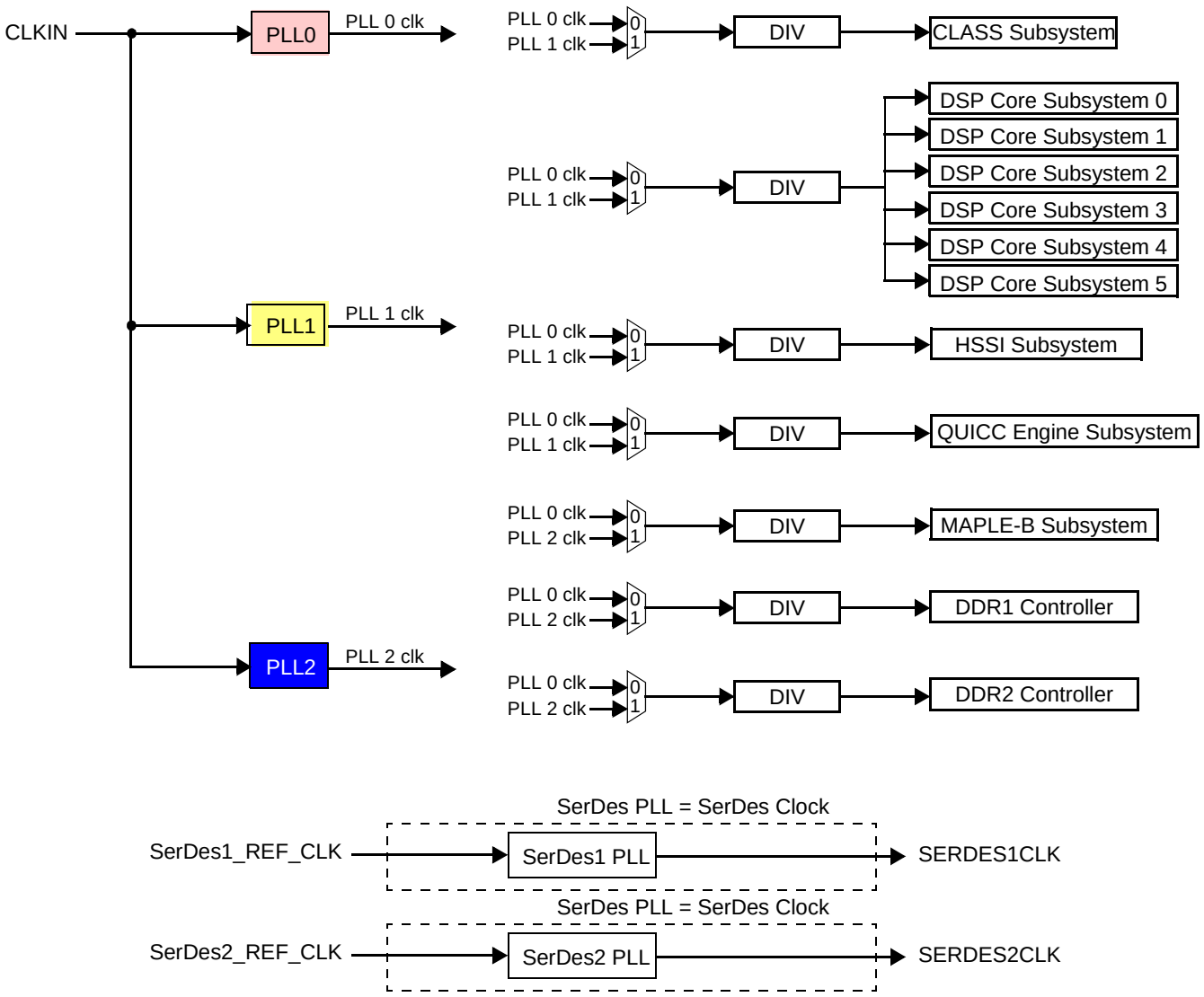
Clocks 7

The clock circuits contains five PLLs:

- Three PLLs driven from a single CLKIN signal generated by a crystal-based oscillator that generate the clocks for the DSP core subsystems, the internal CLASS buses, the RapidIO controller, the QUICC Engine subsystem, the MAPLE-B subsystem, the TDM interfaces, internal memory, the DDR-SDRAM memory controllers, and the PCI Express interface.
- Two PLLs that generate clocks for the SerDes interfaces in the HSSI.

7.1 Clock Generation Components and Modes

The clock generation components and clock scheme are shown in **Figure 7-1**.



Note: The source for CLKOUT is selected at reset via the Reset Configuration Word (RCW). See **Chapter 5**, *Reset* for details.

Figure 7-1. MSC8156E Clock Scheme

Each PLL uses its input clock to generate a fast clock that is synchronized to the input clock. The fast clock is distributed to each of the clock dividers to generate the clocks that are distributed to the system blocks. The clock circuits are locked, according to the selected clock mode, when the first stage of the system reset configuration is done (reset configuration is controlled by the RESET block). The clock circuits are initialized after the first phase of the reset configuration, when the low part of the reset configuration word is loaded, according to the selected clock mode.

The MSC8156E clock modes are listed in **Table 7-1**.

Table 7-1. MSC8156E Clock Modes

Mode	CLKIN	PLL0	PLL1	PLL2	CLASS	DSP Core Subsystems	HSSI	QUICC Engine Subsystem	MAPLE-B	DDR1	DDR2
0	100	900	1000	800	500	1000	333	500	450	800	800
1	66.67	800	0	667	400	800	267	400	400	667	667
4	100	900	1000	667	500	1000	333	450	450	667	667
19	100	900	1000	800	450	1000	333	500	450	800	800
21	100	900	1000	667	450	1000	333	450	450	667	667
36	100	900	1000	800	500	1000	333	500	450	800	267
37	66.67	800	0	667	400	800	267	400	400	667	267
39	100	900	1000	667	500	1000	333	450	450	667	222
45	100	900	1000	667	450	1000	333	450	450	667	222

Notes: **1.** The color of each cell, states which PLL drives its clock domain. PLL 0 is red, PLL1 is yellow, and PLL2 is blue with white lettering.

2. In clock modes 1 and 37, PLL1 is not used. In order to save power and reduce noise, this PLL should be disabled by setting bit 7 of RCW low (for RCW details, see **Chapter 5**, *Reset*).

CLK_OUT pin can be driven from either PLL with selection determined by the value of RCWLR[CLKO] (bits 31–30 of the low part of the reset configuration word—for details, see **Chapter 5**, *Reset*). The possible CLK_OUT frequencies are listed in **Table 7-2**.

Table 7-2. MSC8156E CLK_OUT Frequencies

Mode	PLL0	PLL1	PLL2	CLK_OUT from PLL0	CLK_OUT from PLL1	CLK_OUT from PLL2
0	900	1000	800	75	100	80
1	800	0	667	66.67	6.7	66.67
4	900	1000	667	75	100	66.67
19	900	1000	800	75	100	80
21	900	1000	667	75	100	66.67
36	900	1000	800	75	100	80
37	800	0	667	66.67	6.7	66.67
39	900	1000	667	75	100	66.67
45	900	1000	667	75	100	66.67

7.2 Programming Model

The registers covered in this section are as follows:

- System Clock Control Register (SCCR), page 7-4.
- Clock General Purpose Register 0 (CLK_GRP0), page 7-5.

Note: The clock registers use a base address of: 0xFFF24000.

7.2.1 System Clock Control Register (SCCR)

SCCR System Clock Control Register Offset 0x000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	CLASS DIS	CORE 3-0DIS	CORE 5-4DIS	HSSI DIS	QEDIS	MAPLE DIS	DDR1 DIS	DDR2 DIS	—							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SCCR can be used to shut down the clock for some of the clock domains. This register can only be reset by a power-on reset. **Table 7-3** defines the SCCR bit fields.

Table 7-3. SCCR Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
CLASSDIS 15	0	**CLASS Clock Domain Disable** Used to disable the CLASS clock domain to conserve power.	0 CLASS clock domain enabled. 1 CLASS clock domain disabled.
CORE3-0 DIS 14	0	**Core 3–0 Clock Domain Disable** Used to disable the Core 3–0 clock domain to conserve power.	0 Core 3–0 clock domain enabled. 1 Core 3–0 clock domain disabled.
CORE5-4 DIS 13	0	**Core 5–4 Clock Domain Disable** Used to disable the Core 5–4 clock domain to conserve power.	0 Core 5–4 clock domain enabled. 1 Core 5–4 clock domain disabled.
HSSIDIS 12	0	**HSSI Clock Domain Disable** Used to disable the HSSI clock domain to conserve power.	0 HSSI clock domain enabled. 1 HSSI clock domain disabled.
QEDIS 11	0	**QUICC Engine Clock Domain Disable** Used to disable the QUICC Engine subsystem clock domain to conserve power.	0 QUICC Engine clock domain enabled. 1 QUICC Engine clock domain disabled.
MAPLEDIS 10	0	**MAPLE-B Clock Domain Disable** Used to disable the MAPLE-B clock domain to conserve power.	0 MAPLE-B clock domain enabled. 1 MAPLE-B clock domain disabled.

Table 7-3. SCCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DDR1DIS 9	0	**DDR1 Clock Domain Disable** Used to disable the DDR1 controller clock domain to conserve power.	0 DDR1 clock domain enabled. 1 DDR1 clock domain disabled.
DDR2DIS 8	0	**DDR2 Clock Domain Disable** Used to disable the DDR2 controller clock domain to conserve power.	0 DDR2 clock domain enabled. 1 DDR2 clock domain disabled.
— 7–0	0	Reserved. Write to zero for future compatibility.	

7.2.2 Clock General Purpose Register 0 (CLK_GPR0)

CLK_GPR0 Clock General Purpose Register 0 Offset 0x004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
Reset	Determined by MODCK															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—										RPTE					
Type	R										R/W					
Reset	Determined by MODCK															

The CLK_GPR0 is used to set the RapidIO prescale value to yield the 8 MHz clock required for event timing. **Table 7-4** defines the CLK_GPR0 bit fields.

Table 7-4. CLK_GPR0 Bit Descriptions

Name	Reset	Description	Settings
— 31–6	MODCK value	Reserved.	
RPTE 5–0	MODCK value	**RapidIO Prescaler for Timed Event Clock** This value is used to scale the OCN clock (which is equal to the HSSI clock that is derived from the MODCK settings; see **Table 7-1**) to yield an 8 MHz clock used by the RapidIO subsystem to calculate different event times. The value to enter into this field is computed by the formula: (ocn_clk_freq/8 MHz) – 1, rounded to the nearest whole value (ocn_clk_freq = HSSI clock frequency). The default is selected by the MODCK settings at power-up reset.	100000 For MODCK 1 and 37. 101001 For MODCK 0, 4, 21, 36, 39, 45 All other values reserved.

General Configuration Registers 8

The MSC8156E device includes a general configuration block that includes fifty-six 32-bit registers. This block provides sets of control and status registers for modules in the device that do not include their own control and status registers.

8.1 Programming Model

The general configuration registers include the following:

- General Interrupt Enable Register 3 for Cores 0–5 (GIER3_[0–5]), see **page 8-38**
- General Interrupt Register 5 (GIR5), see **page 8-39**
- General Interrupt Enable Register 5 for Cores 0–5 (GIER5_[0–5]), see **page 8-41**
- General Control Register 11 (GCR11), see **page 8-42**
- General Control Register 12 (GCR12), see **page 8-43**
- DMA Request0 Control Register (GCR_DREQ0), see **page 8-45**
- DMA Request1 Control Register (GCR_DREQ1), see **page 8-49**
- DMA Done Control Register (GCR_DDONE), see **page 8-53**
- DDR1 General Configuration Register (DDR1_GCR), see **page 8-56**
- DDR2 General Configuration Register (DDR2_GCR), see **page 8-57**
- Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR), see **page 8-58**

Note: The base address for the general configuration registers is: 0xFFF28000.

8.2 Detailed Register Descriptions

8.2.1 General Configuration Register 1 (GCR1)

GCR1 General Configuration Register 1 **Offset 0x00**

Bit	31	30	29	28	27	26	25	24
	—							
Type	R			R/W				
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							UART_STOP
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—	DDR2_PIPE_LMT					DDR1_PIPE_LMT	
Type	R/W							
Reset	0	1	0	0	0	0	1	0

Bit	7	6	5	4	3	2	1	0
	DDR1_PIPE_LMT			TDM_PIPE_LMT				
Type	R/W							
Reset	0	0	0	1	0	0	0	0

GCR1 configures various general functions for the MSC8156E device. **Table 8-1** lists the GCR1 bit field descriptions.

Table 8-1. GCR1 Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to 0 for future compatibility.	
UART_STOP 16	0	**UART Stop** Stops the UART clock.	0 Normal operation. 1 UART clock stopped.
— 15	0	Reserved. Write to 0 for future compatibility.	
DDR2_PIPE_LMT 14–10	10000	**DDR2 Pipeline Limit** Specifies the DDR2 complex pipeline depth.	
DDR1_PIPE_LMT 9–5	10000	**DDR1 Pipeline Limit** Specifies the DDR1 complex pipeline depth.	
TDM_PIPE_LMT 4–0	10000	**TDM Pipeline Limit** Specifies the TDM complex pipeline depth.	

8.2.2 General Configuration Register 2 (GCR2)

GCR2 General Configuration Register 2 **Offset 0x04**

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							DMA_DBG
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—		CORE5_STP_EN	CORE4_STP_EN	CORE3_STP_EN	CORE2_STP_EN	CORE1_STP_EN	CORE0_STP_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		CORE5_DBG_REQ	CORE4_DBG_REQ	CORE3_DBG_REQ	CORE2_DBG_REQ	CORE1_DBG_REQ	CORE0_DBG_REQ
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR2 configures various general functions for the MSC8156E device. **Table 8-2** lists the GCR2 bit field descriptions.

Table 8-2. GCR2 Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to 0 for future compatibility.	
DMA_DBG 16	0	**DMA Debug Mode Request** When set, initiates a request for the DMA controller to enter Debug mode. See **Section 14.7.17**, *DMA Debug Event Status Register (DMADESR)*, on page 14-43	0 No request. 1 DMA debug request.
— 15–14	0	Reserved. Write to 0 for future compatibility.	
CORE5_STP_EN 13	0	**Core 5 Stop Enable** Enables core 5 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE4_STP_EN 12	0	**Core 4 Stop Enable** Enables core 4 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE3_STP_EN 11	0	**Core 3 Stop Enable** Enables core 3 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE2_STP_EN 10	0	**Core 2 Stop Enable** Enables core 2 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE1_STP_EN 9	0	**Core 1 Stop Enable** Enables core 1subsystem to stop.	0 Stop disabled. 1 Stop enabled.
CORE0_STP_EN 8	0	**Core 0 Stop Enable** Enables core 0 subsystem to stop.	0 Stop disabled. 1 Stop enabled.
— 7–6	0	Reserved. Write to 0 for future compatibility.	
CORE5_DBG_REQ 5	0	**Core 5 Debug Request** Asserts a debug request to core 5.	0 No debug request. 1 Debug request.
CORE4_DBG_REQ 4	0	**Core 4 Debug Request** Asserts a debug request to core 4.	0 No debug request. 1 Debug request.
CORE3_DBG_REQ 3	0	**Core 3 Debug Request** Asserts a debug request to core 3.	0 No debug request. 1 Debug request.
CORE2_DBG_REQ 2	0	**Core 2 Debug Request** Asserts a debug request to core 2.	0 No debug request. 1 Debug request.
CORE1_DBG_REQ 1	0	**Core 1 Debug Request** Asserts a debug request to core 1.	0 No debug request. 1 Debug request.
CORE0_DBG_REQ 0	0	**Core 0 Debug Request** Asserts a debug request to core 0.	0 No debug request. 1 Debug request.

8.2.3 General Status Register 1 (GSR1)

GSR1 General Status Register 1 **Offset 0x08**

Bit	31	30	29	28	27	26	25	24
	—		CORE_WAIT_ACK5	CORE_WAIT_ACK4	CORE_WAIT_ACK3	CORE_WAIT_ACK2	CORE_WAIT_ACK1	CORE_WAIT_ACK0
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—		CORE_STOP_ACK5	CORE_STOP_ACK4	CORE_STOP_ACK3	CORE_STOP_ACK2	CORE_STOP_ACK1	CORE_STOP_ACK0
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—	M3_PU_1	M3_PU_0	MAPLE_PU	—			
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		CORE_DBG_STS5	CORE_DBG_STS4	CORE_DBG_STS3	CORE_DBG_STS2	CORE_DBG_STS1	CORE_DBG_STS0
Type	R							
Reset	0	0	0	0	0	0	0	0

GSR1 reports the status various general functions for the MSC8156E device. **Table 8-3** lists the GSR1 bit field descriptions.

Table 8-3. GSR1 Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to 0 for future compatibility.	
CORE_WAIT_ACK5 29	0	**Core Wait Acknowledge 5** Reflects whether core 5 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE_WAIT_ACK4 28	0	**Core Wait Acknowledge 4** Reflects whether core 4 subsystem is in Wait state	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE_WAIT_ACK3 27	0	**Core Wait Acknowledge 3** Reflects whether core 3 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE_WAIT_ACK2 26	0	**Core2 Wait Acknowledge 2** Reflects whether core 2 subsystem is in Wait state	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE_WAIT_ACK1 25	0	**Core1 Wait Acknowledge 1** Reflects whether core 1 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.
CORE_WAIT_ACK0 24	0	**Core Wait Acknowledge 0** Reflects whether core 0 subsystem is in Wait state.	0 Core subsystem not in Wait state. 1 Core subsystem in Wait state.

Table 8-3. GSR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 23–22	0	Reserved. Write to 0 for future compatibility.	
CORE_STOP_ACK5 21	0	**Core Stop Acknowledge 5** Reflects whether core 5 subsystem is in Stop state.	0 Core subsystem not in Stop state. 1 Core subsystem in Stop state.
CORE_STOP_ACK4 20	0	**Core Stop Acknowledge 4** Reflects whether core 4 subsystem is in Stop state	0 Core subsystem not in Stop state. 1 Core subsystem in Stop state.
CORE_STOP_ACK3 19	0	**Core Stop Acknowledge 3** Reflects whether core 3 subsystem is in Stop state.	0 Core subsystem not in Stop state. 1 Core subsystem in Stop state.
CORE_STOP_ACK2 18	0	**Core Stop Acknowledge 2** Reflects whether core 2 subsystem is in Stop state	0 Core subsystem not in Stop state. 1 Core subsystem in Stop state.
CORE_STOP_ACK1 17	0	**Core Stop Acknowledge 1** Reflects whether core 1 subsystem is in Stop state.	0 Core subsystem not in Stop state. 1 Core subsystem in Stop state.
CORE_STOP_ACK0 16	0	**Core Stop Acknowledge 0** Reflects whether core 0 subsystem is in Stop state.	0 Core subsystem not in Stop state. 1 Core subsystem in Stop state.
— 15	0	Reserved. Write to 0 for future compatibility.	
M3_PU_1 14	0	**M3 Power Up Second Half** Reflects the M3 512 KB power up status.	0 512 KB M3 (2nd half) powered down. 1 512 KB M3 (2nd half) powered up.
M3_PU_0 13	0	**M3 Power Up First Half** Reflects the M3 512 KB power up status.	0 512 KB M3 (1st half) powered down. 1 512 KB M3 (1st half) powered up.
MAPLE_PU 12	0	**MAPLE Power Up** Reflects the MAPLE block power status.	0 MAPLE powered down. 1 MAPLE powered up.
— 11–6	0	Reserved. Write to 0 for future compatibility.	
CORE_DBG_STS5 5	0	**Core Debug Status 5** Reflects the mode of core 5.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS4 4	0	**Core Debug Status 4** Reflects the mode of core 4.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS3 3	0	**Core Debug Status 3** Reflects the mode of core 3.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS2 2	0	**Core Debug Status 2** Reflects the mode of core 2.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS1 1	0	**Core Debug Status 1** Reflects the mode of core 1.	0 Not in Debug mode. 1 In Debug mode.
CORE_DBG_STS0 0	0	**Core0 Debug Status 0** Reflects the mode of core 0.	0 Not in Debug mode. 1 In Debug mode.

8.2.4 High Speed Serial Interface Status Register (HSSI_SR)

HSSI_SR High Speed Serial Interface Status Register **Offset 0x0C**

Bit	31	30	29	28	27	26	25	24
	—		SERDES2_ PD			DMA1_PD	SRIO1_PD	SERDES1_ PD
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SERDES1_PD		DMA0_PD	SRIO0_PD	PEX_PD	OCN_PD	RMU_PD	SRIO1_STOP _ACK
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SRIO0_STOP _ACK	—			SERDES2_ RST_DONE	SERDES2_ PD	SRIO_1_ IDLE	SRIO_1_OB_I DLE
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SERDES1_ RST_DONE	SERDES1_ PD	SRIO_0_ IDLE	SRIO_0_OB_I DLE	PEX_OB_ IDLE	PEX_IDLE	—	RMU_IDLE
Type	R							
Reset	0	0	0	0	0	0	0	0

HSSI_SR controls part of the SerDes operation for the MSC8156E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-4** lists the HSSI_SR bit field descriptions.

Table 8-4. HSSI_SR Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to 0 for future compatibility.	
SERDES2_PD 29–27	000	**SERDES2 Power Down Status** Indicates the power status of the SERDES2 port. Can be powered down by the Reset Control Word or GCR control.	000 Power up. 111 Power down. All other values reserved.
DMA1_PD 26	0	**DMA1 Power Down Status** Indicates the power status of DMA1. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
SRIO1_PD 25	0	**Serial RapidIO Interface 1 Power Down Status** Indicates the power status of the serial RapidIO interface 1. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
SERDES1_PD 24–22	000	**SERDES1 Power Down Status** Indicates the power status of the SERDES2 port. Can be powered down by the Reset Control Word or GCR control.	000 Power up. 111 Power down. All other values reserved.

Table 8-4. HSSI_SR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA0_PD 21	0	**DMA0 Power Down Status** Indicates the power status of DMA0. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
SRIO0_PD 20	0	**Serial RapidIO Interface 0 Power Down Status** Indicates the power status of the serial RapidIO interface 0. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
PEX_PD 19	0	**PCI Express Power Down Status** Indicates the power status of the PCI Express interface. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
OCN_PD 18	0	**OCN Fabric Power Down Status** Indicates the power status of the OCN fabric. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
RMU_PD 17	0	**RapidIO Messaging Unit Power Down Status** Indicates the power status of the RapidIO Messaging Unit. Can be powered down by the Reset Control Word or GCR control.	0 Power up. 1 Power down.
SRIO1_STOP_ACK 16	0	**Serial RapidIO Interface 1 Stop Acknowledge Status** Indicates the serial RapidIO interface 1 Stop Acknowledge status.	0 Not stopped. 1 Stop ACK issued.
SRIO0_STOP_ACK 15	0	**Serial RapidIO Interface 0 Stop Acknowledge Status** Indicates the serial RapidIO interface 0 Stop Acknowledge status.	0 Not stopped. 1 Stop ACK issued.
— 14–12	0	Reserved. Write to 0 for future compatibility.	
SERDES2_RST_DONE 11	0	**SERDES2 Reset Done** Indicates whether the SERDES2 has completed the reset sequence. **Note:** Although the reset value is 0, the bit will change to 1 within 160 µs after reset depending on whether the SerDes port is used.	0 Reset not complete. 1 Reset complete.
SERDES2_PD 10	0	**SERDES2 Power Down Status** Indicates the power status of the SERDES2 PHY.	0 Power up. 1 Power down (PLL is not working).
SRIO1_IDLE 9	0	**SRIO1 Idle** Indicates whether the SRIO1 unit is idle, that is, no transactions in progress.	0 Active. 1 Idle.
SRIO1_OB_IDLE 8	0	**SRIO1 Outbound Idle** Indicates whether the SRIO1 outbound activity is idle, that is, no outbound transactions in progress.	0 Active. 1 Idle.
SERDES1_RST_DONE 7	0	**SERDES1 Reset Done** Indicates whether the SERDES1 has completed the reset sequence. **Note:** Although the reset value is 0, the bit will change to 1 within 160 µs after reset depending on whether the SerDes port is used.	0 Reset not complete. 1 Reset complete.

Table 8-4. HSSI_SR Bit Descriptions (Continued)

Name	Reset	Description	Settings
SERDES1_PD 6	0	**SERDES1 Power Down Status** Indicates the power status of the SERDES1 PHY.	0 Power up. 1 Power down (PLL is not working).
SRIO0_IDLE 5	0	**SRIO0 Idle** Indicates whether the SRIO0 unit is idle, that is, no transactions in progress.	0 Active. 1 Idle.
SRIO0_OB_IDLE 4	0	**SRIO0 Outbound Idle** Indicates whether the SRIO0 outbound activity is idle, that is, no outbound transactions in progress.	0 Active. 1 Idle.
PEX_OB_IDLE 3	0	**PCI Express Outbound Idle** Indicates whether the PCI Express outbound activity is idle, that is, no outbound transactions in progress.	0 Active. 1 Idle.
PEX_IDLE 2	0	**PCI Express Idle** Indicates whether the PCI Express is idle, that is, no transactions in progress.	0 Active. 1 Idle.
— 1	0	Reserved. Write to 0 for future compatibility.	
RMU_IDLE 0	0	**RMU Idle** Indicates whether the RMU is idle, that is, no transactions in progress.	0 Active. 1 Idle.

8.2.5 DDR General Control Register (DDR_GCR)

DDR_GCR DDR General Control Register **Offset 0x10**

Bit	31	30	29	28	27	26	25	24
	—							DDR2_GCR_VSEL
Type	R				R/W			
Reset	0	0	0	0	1	1	0	1

Bit	23	22	21	20	19	18	17	16
	—		DDR2_COP_TERMSEL_OVERRIDE_VALUE			DDR2_COP_TERMSEL_OVERRIDE_EN	DDR2_DISABLE_BIT_DESKEW	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							DDR1_GCR_VSEL
Type	R/W							
Reset	0	0	0	0	1	1	0	1

Bit	7	6	5	4	3	2	1	0
	—		DDR1_COP_TERMSEL_OVERRIDE_VALUE			DDR1_COP_TERMSEL_OVERRIDE_EN	DDR1_DISABLE_BIT_DESKEW	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

DDR_GCR controls the DDR operation the MSC8156E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-5** lists the DDR_GCR bit field descriptions.

Table 8-5. DDR_GCR Bit Descriptions

Name	Reset	Description	Settings
— 31–25	0	Reserved. Write to 0 for future compatibility.	
DDR2_GCR_VSEL 24	1	**DDRC2 Voltage Select** Indicates the type of memory used.	0 DDR3. 1 DDR2.
— 23–22	0	Reserved. Write to 0 for future compatibility.	
DDR2_COP_TERMSEL_OVERRIDE_VALUE 21–19	0	**DDRC2 Termination Select Override Value** Sets the value for the termination select override.	
DDR2_COP_TERMSEL_OVERRIDE_EN 18	0	**DDRC2 Termination Select Override Enable** Disables/enables the termination select override.	0 Disable 1 Enable.

Table 8-5. DDR_GCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DDR2_DISABLE_BIT_DESKEW 17	0	**DDRC2 Disable Per-Bit Deskew** Disables/enables per-bit deskew calibration initialization.	0 Enable. 1 Disable.
— 16–9	0	Reserved. Write to 0 for future compatibility.	
DDR1_GCR_VSEL 8	1	**DDRC1 Voltage Select** Indicates the type of memory used.	0 DDR3. 1 DDR2.
— 7–6	0	Reserved. Write to 0 for future compatibility.	
DDR1_COP_TERMSEL_OVERRIDE_VALUE 5–3	0	**DDRC1 Termination Select Override Value** Sets the value for the termination select override.	
DDR1_COP_TERMSEL_OVERRIDE_EN 2	0	**DDRC1 Termination Select Override Enable** Disables/enables the termination select override.	0 Disable 1 Enable.
DDR1_DISABLE_BIT_DESKEW 1	0	**DDRC1 Disable Per-Bit Deskew** Disables/enables per-bit deskew calibration initialization.	0 Enable. 1 Disable.
— 0	0	Reserved. Write to 0 for future compatibility.	

8.2.6 High Speed Serial Interface Control Register 1 (HSSI_CR1)

HSSI_CR1 High Speed Serial Interface Control Register 1 **Offset 0x14**

Bit	31	30	29	28	27	26	25	24
	SERDES2_ STOP	SERDES2_ CB_ PD	SERDES2_ ISOR_PD	SERDES2_ DOZE	—		SRIO1_DOZE	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SRIO1_PD	—	SRIO1_ECC_ D	SERDES1_ CB_ PD	SERDES1_ ISOR_PD	SERDES1_ DOZE	—	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SRIO0_DOZE	—	SRIO0_PD	—	—	SERDES1_ STOP	PEX_PD	PEX_DOZE
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		OCN_PD	RMU_PD	RMU_DOZE	—	SERDES2_ CONFIG_ DISABLE	SERDES1_ CONFIG_ DISABLE
Type	R/W							
Reset	0	0	0	0	0	0	1	1

HSSI_CR1 controls various functions within the SerDes block for the MSC8156E device. The register is reset on a hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-6** lists the HSSI_CR1 bit field descriptions.

Table 8-6. HSSI_CR1 Bit Descriptions

Name	Reset	Description	Settings
SERDES2_STOP 31	0	**SERDES2 PHY Stop** Holds the SERDES2 PHY in reset.	0 PHY not stopped. 1 PHY stopped.
SERDES2_CB_PD 30	0	**SERDES2 Control Block Power Down** Shuts down SERDES2 control block ring power.	0 Power up. 1 Power down.
SERDES2_ISOR_PD 29	0	**SERDES2 Isolation Ring Power Down** Shuts down SERDES2 isolation ring power.	0 Power up. 1 Power down.
SERDES2_DOZE 28	0	**SERDES2 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the SerDes port is stopped.	0 Normal operation. 1 Doze mode.
— 27–26	0	Reserved. Write to 0 for future compatibility.	

Table 8-6. HSSI_CR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
SRIO1_DOZE 25	0	**Serial RapidIO Interface 1 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the Serial RapidIO interface clock is stopped.	0 Normal operation. 1 Doze mode.
— 24	0	Reserved. Write to 0 for future compatibility.	
SRIO1_PD 23	0	**Serial RapidIO Interface 1 Power Down** Shuts down serial RapidIO interface 1 power.	0 Power up. 1 Power down.
— 22–21	0	Reserved. Write to 0 for future compatibility.	
SERDES1_CB_PD 20	0	**SERDES1 Control Block Power Down** Shuts down SERDES1 control block ring power.	0 Power up. 1 Power down.
SERDES1_ISOR_PD 19	0	**SERDES1 Isolation Ring Power Down** Shuts down SERDES1 isolation ring power.	0 Power up. 1 Power down.
SERDES1_DOZE 18	0	**SERDES1 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the SerDes port is stopped.	0 Normal operation. 1 Doze mode.
— 17–16	0	Reserved. Write to 0 for future compatibility.	
SRIO0_DOZE 15	0	**Serial RapidIO Interface 0 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the Serial RapidIO clock is stopped.	0 Normal operation. 1 Doze mode.
— 14	0	Reserved. Write to 0 for future compatibility.	
SRIO0_PD 13	0	**Serial RapidIO Interface 0 Power Down** Shuts down serial RapidIO interface 0 power.	0 Power up. 1 Power down.
— 12–11	0	Reserved. Write to 0 for future compatibility.	
SERDES1_STOP 10	0	**SERDES1 PHY Stop** Holds the SERDES1 PHY in reset.	0 PHY not stopped. 1 PHY stopped.
PEX_PD 9	0	**PCI Express Power Down** Shuts down PCI Express power.	0 Power up. 1 Power down.
PEX_DOZE 8	0	**PCI Express Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the PCI Express clock is stopped.	0 Normal operation. 1 Doze mode.

Table 8-6. HSSI_CR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 7–6	0	Reserved. Write to 0 for future compatibility.	
OCN_PD 5	0	**OCN Power Down** Shuts down OCN fabric power.	0 Power up. 1 Power down.
RMU_PD 4	0	**RMU Power Down** Shuts down RMU power.	0 Power up. 1 Power down.
RMU_DOZE 3	0	**RMU Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the RMU is stopped.	0 Normal operation. 1 Doze mode.
—2	0	Reserved. Write to 0 for future compatibility.	
SERDES2_CONFIG_DISABLE 1	1	**SERDES2 SRDS2CR2 Override** Enables/disables SERDES2 SRDS2CR2. When enabled (1), overrides the values in SRDS2CR2[X3SA/X3SB/X3SE/X3SF] and forces them to be 1. This forces all four SerDes lanes to be tri-stated. When disabled (0), there is no override and the values defined for each channel by SRDS2CR2[X3SA/X3SB/X3SE/X3SF] are used to determine whether the port is in a normal operation mode or tri-stated. The value of this bit has no effect on any other fields in SRDS2CR2 and only overrides the specified fields.	0 Disabled. 1 Enabled.
SERDES1_CONFIG_DISABLE 0	1	**SERDES1 SRDS1CR2 Override** Enables/disables SERDES1 SRDS1CR2. When enabled (1), overrides the values in SRDS1CR2[X3SA/X3SB/X3SE/X3SF] and forces them to be 1. This forces all four SerDes lanes to be tri-stated. When disabled (0), there is no override and the values defined for each channel by SRDS1CR2[X3SA/X3SB/X3SE/X3SF] are used to determine whether the port is in a normal operation mode or tri-stated. The value of this bit has no effect on any other fields in SRDS1CR2 and only overrides the specified fields.	0 Disabled. 1 Enabled.

Note: Doze mode is a special mode used by the MSC8156E to allow register reads/writes to continue and be acknowledged without changing or reporting any register contents while the peripheral clocking is stopped. Processing of reads/writes can continue after the peripheral has entered power down mode. Without Doze mode, the device could hang up waiting for a response from the peripheral. This mode permits acknowledgement of the read or write, but does not read or write any meaningful data.

8.2.7 High Speed Serial Interface Control Register 2 (HSSI_CR2)

HSSI_CR2 High Speed Serial Interface Control Register 2 **Offset 0x18**

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—					MAG2SB_STOP	—	RMU_COL_D
Type	R			R/W				
Reset	0	0	0	0	0	0	0	0

HSSI_CR2 controls various functions within the SerDes block for the MSC8156E device. The register is reset on a hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-7** lists the HSSI_CR2 bit field descriptions.

Table 8-7. HSSI_CR2 Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to 0 for future compatibility.	
MAG2SB_STOP 2	0	**MBus to SBus Stop** Stops the master bus to slave bus bridge. The MBus is the internal bus controlled by internal master devices. The SBus is used to access internal registers. This bridge is the interface between the MBus and the SBus. To prevent. lockup if a master device read/writes a peripheral register when the clock is stopped, the MBus completes the access, but read data is invalid and no data is written. This bit can be used to determine if read data is valid or if the write occurred.	0 Not stopped. 1 Stopped.
— 1	0	Reserved. Write to 0 for future compatibility.	
RMU_COL_D 0	0	**RapidIO Messaging Unit Collision Disable** Enables/disables the RMU collision detection.	0 Collision detection enabled. 1 Collision detection disabled.

8.2.8 QUICC Engine Control Register (QECR)

QECR QUICC Engine Control Register **Offset 0x1C**

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—				ENET_SGMII_MODE1	ENET_SGMII_MODE0	—	
Type	R				R/W			
Reset	0	0	0	0	0	0	0	0

QECR controls various functions within the QUICC Engine module for the MSC8156E device. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-8** lists the QECR bit field descriptions.

Table 8-8. QECR Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to 0 for future compatibility.	
ENET_SGMII_MODE1 3	0	**Selects GMII or RGMII for Ethernet Controller 2** Selects SGMII mode for Ethernet controller 2.	0 RGMII selected. 1 SGMII selected.
ENET_SGMII_MODE0 2	0	**Selects GMII or RGMII for Ethernet Controller 1** Selects SGMII mode for Ethernet controller 1.	0 RGMII1 selected. 1 SGMII1 selected.
— 1–0	0	Reserved. Write to 0 for future compatibility.	

8.2.9 GPIO Pull-Up Enable Register (GPUER)

GPUER GPIO Pull-Up Enable Register **Offset 0x20**

Bit	31	30	29	28	27	26	25	24
	PUE_B31	PUE_B30	PUE_B29	PUE_B28	PUE_B27	PUE_B26	PUE_B25	PUE_B24
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PUE_B23	PUE_B22	PUE_B21	PUE_B20	PUE_B19	PUE_B18	PUE_B17	PUE_B16
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PUE_B15	PUE_B14	PUE_B13	PUE_B12	PUE_B11	PUE_B10	PUE_B9	PUE_B8
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PUE_B7	PUE_B6	PUE_B5	PUE_B4	PUE_B3	PUE_B2	PUE_B1	PUE_B0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GPUER enables/disables the individual GPIO pull-up resistors. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-9** lists the GPUER bit field descriptions.

Table 8-9. GPUER Bit Descriptions

Name	Reset	Description	Settings
PUE_B[31–0] 31–0	0	**Pull-Up Enable 31–0** Each bit in this field enables/disables the GPIO pull-up resistor corresponding to the bit index number.	0 Pull-up input is enabled. 1 Pull-up input is disabled.

8.2.10 GPIO Input Enable Register (GIER)

GIER GPIO Input Enable Register **Offset 0x24**

Bit	31	30	29	28	27	26	25	24
	IE31	IE30	IE29	IE28	IE27	IE26	IE25	IE24
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	IE23	IE22	IE21	IE20	IE19	IE18	IE17	IE16
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER enables/disables the individual GPIO signals. The register is reset on a hard reset. Write accesses can only be performed in Supervisor mode. **Table 8-10** lists the GIER bit field descriptions.

Table 8-10. GIER Bit Descriptions

Name	Reset	Description	Settings
IE[31–0] 31–0	0	**Input Enable 31–0** Each bit in this field enables/disables the individual GPIO corresponding to the bit index number.	0 Input is disabled. 1 Input is enabled.

8.2.11 System Part and Revision ID Register (SPRIDR)

SPRIDR System Part and Revision ID Register **Offset 0x28**

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

SPRIDR provides information about the device and revision numbers. **Table 8-11** lists the SPRIDR bit field descriptions.

Table 8-11. SPRIDR Bit Descriptions

Name	Reset	Description
PARTID 31–16	0x830A	**Part Identification** Mask-programmed with a code corresponding to the device number.
REVID 15–0	0x0000	**Revision Identification** Mask-programmed with a code corresponding to the revision number of the part identified by the PARTID value.

8.2.12 General Control Register 4 (GCR4)

GCR4 General Control Register 4 Offset 0x30

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—				UCC3RCLKID		UCC3TCLKID	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	UCC3CLKOD		UCC3RXDD		UCC3TXDD		UCC1RCLKID	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	UCC1TCLKID		UCC1CLKOD		UCC1RXDD		UCC1TXDD	
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR4 controls the delay lines for UCC1 and UCC3. The register is reset on a Hard reset. Write accesses can only be performed in Supervisor mode.

The MSC8156E Data Sheet includes recommended default values for this register to use with a standard RGMII PHY device. **AN3811** *Using GCR4 to Adjust Ethernet Timing in MSC8144 DSPs* (available under NDA) provides guidelines for adjusting GCR4 values for specific applications, if required. Although this application note is directed toward designs using the MSC8144 DSP, the procedures used to adjust GCR4 apply to the MSC8156E DSP.

Table 8-8 lists the GCR4 bit field descriptions.

Table 8-12. GCR4 Bit Descriptions

Name	Reset	Description	Settings
— 31–20	0	Reserved. Write to 0 for future compatibility.	
UCC3RCLKID 19–18	0	**UCC3 RX Clock In Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC3TCLKID 17–16	0	**UCC3 TX Clock In Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.

Table 8-12. GCR4 Bit Descriptions (Continued)

Name	Reset	Description	Settings
UCC3CLKOD 15–14	0	**UCC3 Clock Out Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC3RXDD 13–12	0	**UCC3 RX Data Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC3TXDD 11–10	0	**UCC3 TX Data Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC1RCLKID 9–8	0	**UCC1 RX Clock In Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC1TCLKID 7–6	0	**UCC1 TX Clock In Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC1CLKOD 5–4	0	**UCC1 Clock Out Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC1RXDD 3–2	0	**UCC1 RX Data Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
UCC1TXDD 1–0	0	**UCC1 TX Data Delay** Adds a delay to the specified signal.	00 No delay. 01 One delay unit. 10 Two delay units. 11 Three delay units.
Note: The clock for the delay unit is the TX clock.			

8.2.13 General Control Register 5 (GCR5)

GCR5 General Control Register 5 Offset 0x34

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—			PEX_IRQ_OUT	OCNDMA1_POWER DOWN	OCNDMA1_DOZE	OCNDMA1_STOP	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—				OCNDMA0_POWER DOWN	OCNDMA0_DOZE	OCNDMA0_STOP	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR5 performs various control functions. All bits are cleared on reset.

Table 8-13. GCR5 Bit Descriptions

Name	Reset	Description	Settings
— 31–13	0	Reserved. Write to zero for future compatibility.	
PEX_IRQ_OUT 12	0	**PCI Express Message Signal Interrupt** Triggers the PCI Express message signal interrupt.	0 No PCI Express message. 1 PCI Express message interrupt.
OCNDMA1_POWER DOWN 11	0	**OCNDMA 1 Complex Power Down** Makes the OCNDMA1 complex power down.	0 OCNDMA1 powered up. 1 OCNDMA1 power down (Stop ACK).
OCNDMA1_DOZE 10	0	**OCNDMA 1 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the OCNDMA1 is stopped.	0 Normal operation. 1 Doze mode.

Table 8-13. GCR5 Bit Descriptions (Continued)

Name	Reset	Description	Settings
OCNDMA1_STOP 9	0	**OCNDMA 1 Stop** Makes the OCNDMA1 enter Stop mode.	0 OCNDMA1 normal operation. 1 OCNDMA1 Stop mode.
— 8–4	0	Reserved. Write to zero for future compatibility.	
OCNDMA0_POWER DOWN 3	0	**OCNDMA 0 Complex Power Down** Drives the ips_wait signal to 0 in preparation for power down to avoid ips transactions becoming stuck.	0 OCNDMA0 powered up. 1 OCNDMA0 power down (Stop ACK).
OCNDMA0_DOZE 2	0	**OCNDMA 0 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the OCNDMA0 is stopped.	0 Normal operation. 1 Doze mode.
OCNDMA0_STOP 1	0	**OCNDMA 0 Stop** Makes the OCNDMA0 enter Stop mode.	0 OCNDMA0 normal operation. 1 OCNDMA0 Stop mode.
— 0	0	Reserved. Write to zero for future compatibility.	

8.2.14 General Status Register 2 (GSR2)

GSR2 General Status Register 2 Offset 0x38

Bit	31	30	29	28	27	26	25	24
	DDR2_IDLE_MEM	DDR2_YMMC_STOP_ACK	DDR1_IDLE_MEM	DDR1_YMMC_STOP_ACK	—			
Type	R							
Reset	1	0	1	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—		CORE_STOP_REQ5	CORE_STOP_REQ4	CORE_STOP_REQ3	CORE_STOP_REQ2	CORE_STOP_REQ1	CORE_STOP_REQ0
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		SCOP_IDLE	—	OCNDMA1_IDLE	—	OCNDMA0_IDL:E	—
Type	R							
Reset	0	0	1	0	1	0	1	0

GSR2 reflects the status of various functions. All bits are cleared on reset.

Table 8-14. GSR2 Bit Descriptions

Name	Reset	Description	Settings
DDR2_IDLE_MEM 31	0	**DDR2 Controller Idle** Reflects the current status of DDR Controller 2.	0 Memory controller 2 active. 1 Memory controller 2 idle.
DDR2_YMMC_STOP_ACK 30	0	**DDR2 Controller Refresh Mode** Reflects the current status of DDR Controller 2 refresh mode.	0 Memory controller 2 not in self-refresh mode. 1 Memory controller 2 in self-refresh mode.
DDR1_IDLE_MEM 29	0	**DDR1 Controller Idle** Reflects the current status of DDR Controller 1.	0 Memory controller 1 active. 1 Memory controller 1 idle.
DDR1_YMMC_STOP_ACK 30	0	**DDR1 Controller Refresh Mode** Reflects the current status of DDR Controller 1 refresh mode.	0 Memory controller 1 not in self-refresh mode. 1 Memory controller 1 in self-refresh mode.
— 27–22	0	Reserved. Write to zero for future compatibility.	

Table 8-14. GSR2 Bit Descriptions (Continued)

Name	Reset	Description	Settings
CORE_STOP_REQ5 21	0	**Core 5 Stop Request** Reflects whether a core stop was requested.	0 Core 5 stop not requested. 1 Core 5 stop requested.
CORE_STOP_REQ4 20	0	**Core 4 Stop Request** Reflects whether a core stop was requested.	0 Core 4 stop not requested. 1 Core 4 stop requested.
CORE_STOP_REQ3 19	0	**Core 3 Stop Request** Reflects whether a core stop was requested.	0 Core 3 stop not requested. 1 Core 3 stop requested.
CORE_STOP_REQ2 18	0	**Core 2 Stop Request** Reflects whether a core stop was requested.	0 Core 2 stop not requested. 1 Core 2 stop requested.
CORE_STOP_REQ1 17	0	**Core 1 Stop Request** Reflects whether a core stop was requested.	0 Core 1 stop not requested. 1 Core 1 stop requested.
CORE_STOP_REQ0 16	0	**Core 0 Stop Request** Reflects whether a core stop was requested.	0 Core 0 stop not requested. 1 Core 0 stop requested.
— 15–6	0	Reserved. Write to zero for future compatibility.	
SCOP_IDLE 5	0	**SEC Idle** Reflects the current status of the SEC block.	0 Active 1 Idle
— 4	0	Reserved. Write to zero for future compatibility.	
OCNDMA1_IDLE 3	0	**OCNDMA1 Idle** Reflects the current status of the OCNDMA1 block.	0 Active 1 Idle
— 2	0	Reserved. Write to zero for future compatibility.	
OCNDMA0_IDLE 1	0	**OCNDMA0 Idle** Reflects the current status of the OCNDMA0 block.	0 Active 1 Idle
— 0	0	Reserved. Write to zero for future compatibility.	

8.2.15 Core Subsystem Slave Port Priority Control Register (TSPPCR)

TSPPCR Core Subsystem Slave Port Priority Control Register Offset 0x3C

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—				SPP_P3_MAP	SPP_P2_MAP	SPP_P1_MAP	SPP_P0_MAP
Type	R				R/W			
Reset	0	0	0	0	1	1	0	0

TSPPCR reflects the priority assigned to the core subsystem slave ports.

Table 8-15. TSPPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0x0000000	Reserved. Write to zero for future compatibility.	
SPP_P3_MAP 3	1	**Slave Port Mapping for Priority 3** Indicates the priority for the core slave port.	0 All transactions with priority 3 are assigned priority 0. 1 All transactions with priority 3 are assigned priority 1.
SPP_P2_MAP 2	1	**Slave Port Mapping for Priority 2** Indicates the priority for the core slave port.	0 All transactions with priority 2 are assigned priority 0. 1 All transactions with priority 2 are assigned priority 1.
SPP_P1_MAP 1	0	**Slave Port Mapping for Priority 1** Indicates the priority for the core slave port.	0 All transactions with priority 1 are assigned priority 0. 1 All transactions with priority 1 are assigned priority 1.
SPP_P0_MAP 0	0	**Slave Port Mapping Priority 0** Indicates the priority for the core slave port.	0 All transactions with priority 0 are assigned priority 0. 1 All transactions with priority 0 are assigned priority 1.

8.2.16 QUICC Engine First External Request Multiplex Register (CPCE1R)

CPCE1R QUICC Engine First External Request Multiplex Register Offset 0x40

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—						QE_INT1_MNG	
Type	R						R/W	
Reset	0	0	0	0	0	0	0	0

CPE1R determines how the first QUICC Engine external request is assigned. All bits are cleared on reset.

Table 8-16. CPE1R Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
QE_INT1_MNG 1–0	0	**QUICC Engine External Request 1 Multiplexing** Indicates how to process the external request.	00 RapidIO interrupt. 01 Reserved. 10 SEC primary interrupt. 11 Reserved.

8.2.17 QUICC Engine Second External Request Multiplex Register (CPCE2R)

CPCE2R QUICC Engine Second External Request Multiplex Register Offset 0x44

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—						QE_INT2_MNG	
Type	R						R/W	
Reset	0	0	0	0	0	0	0	0

CPE2R determines how the second QUICC Engine external request is assigned. All bits are cleared on reset.

Table 8-17. CPE2R Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
QE_INT2_MNG 1–0	0	**QUICC Engine External Request 2 Multiplexing** Indicates how to process the external request.	00 RapidIO interrupt. 01 Reserved. 10 SEC primary interrupt. 11 Reserved.

8.2.18 QUICC Engine Third External Request Multiplex Register (CPCE3R)

CPCE3R QUICC Engine Third External Request Multiplex Register Offset 0x48

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—						QE_INT3_MNG	
Type	R						R/W	
Reset	0	0	0	0	0	0	0	0

CPE3R determines how the third QUICC Engine external request is assigned. All bits are cleared on reset.

Table 8-18. CPE3R Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
QE_INT3_MNG 1–0	0	**QUICC Engine External Request 3 Multiplexing** Indicates how to process the external request.	00 RapidIO interrupt. 01 Reserved. 10 SEC primary interrupt. 11 Reserved.

8.2.19 QUICC Engine Fourth External Request Multiplex Register (CPCE4R)

CPCE4R QUICC Engine Fourth External Request Multiplex Register Offset 0x4C

Bit	31	30	29	28	27	26	25	24
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—						QE_INT4_MNG	
Type	R						R/W	
Reset	0	0	0	0	0	0	0	0

CPE4R determines how the fourth QUICC Engine external request is assigned. All bits are cleared on reset.

Table 8-19. CPE4R Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
QE_INT4_MNG 1–0	0	**QUICC Engine External Request 4 Multiplexing** Indicates how to process the external request.	00 RapidIO interrupt. 01 Reserved. 10 SEC primary interrupt. 11 Reserved.

8.2.20 General Control Register 10 (GCR10)

GCR10 General Control Register 10 Offset 0x74

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	—				MCK1_EN_DDR1	MCK2_EN_DDR1	MCK1_EN_DDR2	MCK2_EN_DDR2
Type	R/W							
Reset	1	1	1	1	1	1	1	1

GCR10 is used to control the DDR controller clocks.

Table 8-20. GCR10 Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0xFFFFFFF	Reserved. Write to zero for future compatibility.	
MCK1_EN_DDR1 3	1	**DDR1 MCK1 Enable** Enables/disables MCK1 for DDR1	0 Disabled. 1 Enabled
MCK2_EN_DDR1 2	1	**DDR1 MCK2 Enable** Enables/disables MCK2 for DDR1	0 Disabled. 1 Enabled
MCK1_EN_DDR2 1	1	**DDR2 MCK1 Enable** Enables/disables MCK1 for DDR2	0 Disabled. 1 Enabled
MCK2_EN_DDR2 0	1	**DDR2 MCK2 Enable** Enables/disables MCK2 for DDR2	0 Disabled. 1 Enabled

8.2.21 General Interrupt Register 1 (GIR1)

GIR1 General Interrupt Register 1 Offset 0x80

Bit	31	30	29	28	27	26	25	24
	SWT7	SWT6	SWT5	SWT4	SWT3	SWT2	SWT1	SWT0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	O2M1_ERR	O2M0_ERR	—	DMA_ERR	CE_IECC	CE_DECC	—	TDM_P0ECC
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TDM3_TERR	TDM3_RERR	TDM2_TERR	TDM2_RERR	TDM1_TERR	TDM1_RERR	TDM0_TERR	TDM0_RERR
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR1 includes the interrupt status of several events that are rare. Those bits are not sticky and only sample the events. The GIR1 is reset by a hard reset event. All bits are cleared on reset.

Table 8-21. GIR1 Bit Descriptions

Name	Reset	Description	Settings
SWT7 31	0	**Software Watchdog Timer 7** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
SWT6 30	0	**Software Watchdog Timer 6** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
SWT5 29	0	**Software Watchdog Timer 5** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
SWT4 28	0	**Software Watchdog Timer 4** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
SWT3 27	0	**Software Watchdog Timer 3** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
SWT2 26	0	**Software Watchdog Timer 2** Reflects the status of the watchdog timer interrupt.See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted

Table 8-21. GIR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
SWT1 25	0	**Software Watchdog Timer 1** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
SWT0 24	0	**Software Watchdog Timer 0** Reflects the status of the watchdog timer interrupt. See **Section 21.3**, *Software Watchdog Timers*.	0 Interrupt not asserted 1 Interrupt asserted
O2M1_ERR 23	0	**O2M 1 Error** Reflects the status of the O2M1 error interrupt. Possible causes for an interrupt include: • Unsupported packet type detected on the OCN outbound interface. • OCN end-of-packet signal is asserted before the expected end-of-transaction. • Data error. Data is corrupted on the O2M bridge.	0 Interrupt not asserted 1 Interrupt asserted
O2M0_ERR 22	0	**O2M 0 Error** Reflects the status of the O2M0 error interrupt. Possible causes for an interrupt include: • Unsupported packet type detected on the OCN outbound interface. • OCN end-of-packet signal is asserted before the expected end-of-transaction. • Data error. Data is corrupted on the O2M bridge.	0 Interrupt not asserted 1 Interrupt asserted
— 21	0	Reserved. Write to zero for future compatibility.	
DMA_ERR 20	0	**DMA Error** Reflects the status of the DMA error interrupt. See **Section 14.7.16**, *DMA Error Register (DMAERR)*.	0 Interrupt not asserted 1 Interrupt asserted
CE_IECC 19	0	**QUICC Engine IRAM Error** Reflects the status of the QUICC Engine IRAM ECC error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
CE_DECC 18	0	**QUICC Engine DRAM Error** Reflects the status of the QUICC Engine DRAM ECC error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 17	0	Reserved. Write to zero for future compatibility.	
TDM_P0ECC 16	0	**TDM Parity Error** Reflects the ORed status of the TDM[0–3] parity error interrupts.	0 Interrupt not asserted 1 Interrupt asserted
— 15–8	0	Reserved. Write to zero for future compatibility.	
TDM3_TERR 7	0	**TDM3 Transmit Error** Reflects the status of the TDM3 transmit error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
TDM3_RERR 6	0	**TDM3 Receive Error** Reflects the status of the TDM3 receive error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
TDM2_TERR 5	0	**TDM2 Transmit Error** Reflects the status of the TDM2 transmit error interrupt.	0 Interrupt not asserted 1 Interrupt asserted

Table 8-21. GIR1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
TDM2_RERR 4	0	**TDM2 Receive Error** Reflects the status of the TDM2 receive error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
TDM1_TERR 3	0	**TDM1 Transmit Error** Reflects the status of the TDM1 transmit error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
TDM1_RERR 2	0	**TDM1 Receive Error** Reflects the status of the TDM1 receive error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
TDM0_TERR 1	0	**TDM0 Transmit Error** Reflects the status of the TDM0 transmit error interrupt.	0 Interrupt not asserted 1 Interrupt asserted
TDM0_RERR 0	0	**TDM0 Receive Error** Reflects the status of the TDM0 receive error interrupt.	0 Interrupt not asserted 1 Interrupt asserted

8.2.22 General Interrupt Enable Register 1 (GIER1_x)

GIER1_0 ‘General Interrupt Enable Register 1 for Cores 0–5 Offset 0x84
GIER1_1 Offset 0x88
GIER1_2 Offset 0x8C
GIER1_3 Offset 0x90
GIER1_4 Offset 0x94
GIER1_5 Offset 0x98

Bit	31	30	29	28	27	26	25	24
	SWT7_EN_n	SWT6_EN_n	SWT5_EN_n	SWT4_EN_n	SWT3_EN_n	SWT2_EN_n	SWT1_EN_n	SWT0_EN_n
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	O2M1_ERR_EN_n	O2M0_ERR_EN_n	—	DMA_ERR_EN_n	CE_IECC_EN_n	CE_DECC_EN_n	—	TDM_P0ECC_EN_n
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TDM3_TERR_EN_n	TDM3_RERR_EN_n	TDM2_TERR_EN_n	TDM2_RERR_EN_n	TDM1_TERR_EN_n	TDM1_RERR_EN_n	TDM0_TERR_EN_n	TDM0_RERr_EN_n
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER1_[0–5] includes interrupt enable bits of for the interrupts defined in GIR1 for cores 0–5. The register is reset by a hard reset event. All bits are cleared by reset. Write accesses to this register can only be performed in supervisor mode.

Table 8-22. GIER1_n Bit Descriptions

Name	Reset	Description	Settings
SWT7_EN_n 31	0	**SWT 7 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT6_EN_n 30	0	**SWT 6 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT5_EN_n 29	0	**SWT 5 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT4_EN_n 28	0	**SWT 4 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT3_EN_n 27	0	**SWT 3 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT2_EN_n 26	0	**SWT 2 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT1_EN_n 25	0	**SWT 1 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
SWT0_EN_n 24	0	**SWT 0 Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
O2M1_ERR_EN_n 23	0	**O2M1 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
O2M0_ERR_EN_n 22	0	**O2M0 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 21	0	Reserved. Write to zero for future compatibility.	
DMA_ERR_EN_n 20	0	**DMA Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CE_IECC_EN_n 19	0	**ECC Error Interrupt of the QUICC Engine IMEM Enable**	0 Interrupt disabled 1 Interrupt enabled
CE_DECC_EN_n 18	0	**ECC Error Interrupt of the QUICC Engine DRAM Enable**	0 Interrupt disabled 1 Interrupt enabled
— 17	0	Reserved. Write to zero for future compatibility.	
TDM_P0ECC_EN_n 16	0	**Parity Error Interrupt of TDM[0–3] Enable**	0 Interrupt disabled 1 Interrupt enabled
— 15–8	0	Reserved. Write to zero for future compatibility.	
TDM3_TER_EN_n 7	0	**TDM3 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 8-22. GIER1_n Bit Descriptions

Name	Reset	Description	Settings
TDM3_RER_EN_n 6	0	**TDM3 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM2_TER_EN_n 5	0	**TDM2 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM2_RER_EN_n 4	0	**TDM2 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM1_TER_EN_n 3	0	**TDM1 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM1_RER_EN_n 2	0	**TDM1 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM0_TER_EN_n 1	0	**TDM0 Transmit Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
TDM0_RER_EN_n 0	0	**TDM0 Receive Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

8.2.23 General Interrupt Register 3 (GIR3)

GIR3 General Interrupt Register 3 Offset 0xA4

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—	DDR2_ERR	DDR1_ERR	MAPLE_ECC_DRAM	MAPLE_ECC_IMEM	MAPLE_ECC_DFT	MAPLE_ECC_FFT	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—			MAPLE_GEN_ERR	—			PM
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—			CLS0_ERR	—		CLS0_WP	CLS0_OV
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIR3 includes interrupt status of some debug/profiling events within MSC8156E. Those bits are not sticky but only sample the events. The GIR3 register is reset by a hard reset event. All bits are cleared on reset.

Table 8-23. GIR3 Bit Descriptions

Name	Reset	Description	Settings
— 31–23	0	Reserved. Write to zero for future compatibility.	
DDR2_ERR 22	0	**DDR2 Error Interrupt** Reflects the status of the interrupt. See **Section 12.6**, *Error Management*.	0 Interrupt not asserted 1 Interrupt asserted
DDR1_ERR 21	0	**DDR1 Error Interrupt** Reflects the status of the interrupt. See **Section 12.6**, *Error Management*.	0 Interrupt not asserted 1 Interrupt asserted
MAPLE_ECC_DRAM 20	0	**MAPLE DRAM Error Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
MAPLE_ECC_IMEM 19	0	**MAPLE IMEM Error Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
MAPLE_ECC_DFT 18	0	**MAPLE DFT ECC Error Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
MAPLE_ECC_FFT 17	0	**MAPLE FFT0 ECC Error Interrupt** Reflects the status of the interrupt.	0 Interrupt not asserted 1 Interrupt asserted
— 16–13	0	Reserved. Write to zero for future compatibility.	
MAPLE_GEN_ERR 12	0	**MAPLE General Error Interrupt** Reflects the status of the interrupt. This is an aggregation of errors that can occur in the MAPLE-B. See **Section 26.3.3.2**, *Interrupts*.	0 Interrupt not asserted 1 Interrupt asserted
— 11–9	0	Reserved. Write to zero for future compatibility.	
PM 8	0	**Performance Monitor Interrupt** Reflects the performance monitor interrupt	0 Interrupt not asserted 1 Interrupt asserted
— 7–5	0	Reserved. Write to zero for future compatibility.	
CLS0_ERR 4	0	**CLASS0 Error Interrupt** Reflects CLASS0 Error Interrupt	0 Interrupt not asserted 1 Interrupt asserted
— 3–2	0	Reserved. Write to zero for future compatibility.	
CLS0_WP 1	0	**CLASS0 Watchpoint Interrupt** Reflects Class0 watchpoint interrupt	0 Interrupt not asserted 1 Interrupt asserted
CLS0_OV 0	0	**CLASS0 Overrun Interrupt** Reflects CLASS0 overrun interrupt	0 Interrupt not asserted 1 Interrupt asserted

8.2.24 General Interrupt Enable Register 3 for Cores 0–3 (GIER3_x)

GIER3_0 General Interrupt Enable Register 3 for Cores 0–3 Offset 0xA8
GIER3_1 Offset 0xAC
GIER3_2 Offset 0xB0
GIER3_3 Offset 0xB4
GIER3_4 Offset 0xB8
GIER3_5 Offset 0xBC

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—	DDR2_ERR_EN_n	DDR1_ERR_EN_n	MAPLE_ECC_DRAM_EN_n	MAPLE_ECC_IMEM_EN_n	MAPLE_ECC_DFT_EN_n	MAPLE_ECC_FFT_EN_n	—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—			MAPLE_GEN_ERR_EN_n	—			PM_EN_n
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—			CLS0_ERR_EN_n	—		CLS0_WP_EN_n	CLS0_OV_EN_n
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER3_[0–5] include interrupt enable bits for cores 0–5 for debug/profiling events within MSC8156E. GIER3_[0–5] are reset by a hard reset event. All bits are cleared on reset. Write accesses to this register can be performed only in supervisor mode.

Table 8-24. GIER3_[0–5] Bit Descriptions

Name	Reset	Description	Settings
— 31–23	0	Reserved. Write to zero for future compatibility.	
DDR2_ERR_EN_n 22	0	**DDR2 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
DDR1_ERR_EN_n 21	0	**DDR1 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
MAPLE_ECC_DRAM_EN_n 20	0	**MAPLE DRAM Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
MAPLE_ECC_IMEM_EN_n 19	0	**MAPLE IMEM Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

Table 8-24. GIER3_[0–5] Bit Descriptions

Name	Reset	Description	Settings
MAPLE_ECC_DFT_EN_n 18	0	**MAPLE FFT1 ECC Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
MAPLE_ECC_FFT_EN_n 17	0	**MAPLE FFT0 ECC Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 13–16	0	Reserved. Write to zero for future compatibility.	
MAPLE_GEN_ERR_EN_n 12	0	**MAPLE General Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 11–9	0	Reserved. Write to zero for future compatibility.	
PM_EN_n 8	0	**Performance Monitor Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 7–5	0	Reserved. Write to zero for future compatibility.	
CLS0_ERR_EN_n 4	0	**CLASS0 Error Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
— 3–2	0	Reserved. Write to zero for future compatibility.	
CLS0_WP_EN_n 1	0	**CLASS0 Watchpoint Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled
CLS0_OV_EN_n 0	0	**CLASS0 Overrun Interrupt Enable**	0 Interrupt disabled 1 Interrupt enabled

8.2.25 General Interrupt Register 5 (GIR5)

GIR5 General Interrupt Register 5 Offset 0xEC

Bit	31	30	29	28	27	26	25	24
	—							
Type	W1C							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	W1C							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	W1C							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—					T4_T5_AE	T2_T3_AE	T0_T1_AE
Type	W1C							
Reset	0	0	0	0	0	0	0	0

GIR5 includes interrupt status of some internal events within MSC8156E. Those bits are sticky and cleared by writing a 1 to the bit. The GIR5 register is reset by a hard reset event. All bits are cleared on reset. Write accesses to this register can only be performed in supervisor mode.

Table 8-25. GIR5 Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
T4_T5_AE 2	0	**Core 4 or Core 5 L2/M2 Access Error Interrupt** Reflects L2/M2 Access Error Interrupt which occurs when the referenced core is in Stop mode. One of the cores tried to access core subsystem 4 or core subsystem 5 M2/L2 memory.	0 Interrupt not asserted 1 Interrupt asserted
T2_T3_AE 1	0	**Core 2 or Core3 L2/M2 Access Error Interrupt** Reflects L2/M2 Access Error Interrupt which occurs when the referenced core is in Stop mode.One of the cores tried to access core subsystem 2 or core subsystem 3 M2/L2 memory.	0 Interrupt not asserted 1 Interrupt asserted
T0_T1_AE 0	0	**Core 0 or Core 1 L2/M2 Access Error Interrupt** Reflects L2/M2 Access Error Interrupt which occurs when the referenced core is in Stop mode.One of the cores tried to access core subsystem 0 or core subsystem 1 M2/L2 memory.	0 Interrupt not asserted 1 Interrupt asserted

8.2.26 General Interrupt Enable Register 5 (GIER5_x)

GIER5_0 General Interrupt Enable Register 5 for Cores 0–5 Offset 0xF0
GIER5_1 Offset 0xF4
GIER5_2 Offset 0xF8
GIER5_3 Offset 0xFC
GIER5_4 Offset 0x100
GIER5_5 Offset 0x104

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—					T4_T5_AE_EN	T2_T3_AE_EN	T0_T1_AE_EN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GIER5_[0–5] include interrupt enable bits for cores 0–5 for interrupts defined by GIR5. GIER5_[0–5] are reset by a hard reset event. All bits are cleared on reset. Write accesses to this register can be performed only in supervisor mode.

Table 8-26. GIER5_[0–5] Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
T4_T5_AE_EN_n 2	0	**Core 4 or Core 5 L2/M2 Access Error Interrupt Enable** Enables/disable the interrupt.	0 Interrupt disabled 1 Interrupt enabled
T2_T3_AE_EN_n 1	0	**Core 2 or Core 3 L2/M2 Access Error Interrupt Enable** Enables/disable the interrupt.	0 Interrupt disabled 1 Interrupt enabled
T0_T1_AE_EN_n 0	0	**Core 0 or Core 1 L2/M2 Access Error Interrupt Enable** Enables/disable the interrupt.	0 Interrupt disabled 1 Interrupt enabled

8.2.27 General Control Register 11 (GCR11)

GCR11 General Control Register 11 Offset 0x110

Bit	31	30	29	28	27	26	25	24
	—							PERIPHER_ SYS_LATE_ ARBITARTION
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—				PERIPHER_ SYS_INIT2_WEIGHT			
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PERIPHER_ SYS_INIT1_WEIGHT				PERIPHER_ SYS_INIT0_WEIGHT			
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR11 controls the initial values for weighted and late arbitration operation of the peripheral system. All bits are cleared on reset.

Table 8-27. GCR11 Bit Descriptions

Name	Reset	Description	Settings
— 31–25	0	Reserved. Write to zero for future compatibility.	
PERIPHER_ SYS_LATE_ ARBITRATION 24	0	**Late Arbitration** Enables/disables late arbitration for the peripheral system.	0 Do not allow late arbitration. 1 Allow late arbitration.
— 23–12	0	Reserved. Write to zero for future compatibility.	
PERIPHER_ SYS_INIT2_ WEIGHT 11–8	0	**Initial TDM Arbitration Weight** Sets the initial weight given to TDM transactions for peripheral system arbitration.	
PERIPHER_ SYS_INIT1_ WEIGHT 7–4	0	**Initial QUICC Engine Module Arbitration Weight** Sets the initial weight given to the QUICC Engine module for peripheral system arbitration.	
PERIPHER_ SYS_INIT0_ WEIGHT 3–0	0	**Initial SEC Arbitration Weight** Sets the initial weight given to the SEC for peripheral system arbitration.	

8.2.28 General Control Register 12 (GCR12)

GCR12 General Control Register 12 Offset 0x114

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							HSSI_SYS_LATE_ARBITARTION
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HSSI_SYS_INIT1_WEIGHT				HSSI_SYS_INIT0_WEIGHT			
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR12 controls the initial values for weighted and late arbitration operation of the HSSI CLASS. All bits are cleared on reset.

Table 8-28. GCR12 Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
HSSI_SYS_LATE_ARBITRATION 16	0	**Late Arbitration** Enables/disables late arbitration for the HSSI system.	0 Do not allow late arbitration. 1 Allow late arbitration.
— 15–8	0	Reserved. Write to zero for future compatibility.	

Table 8-28. GCR12 Bit Descriptions (Continued)

Name	Reset	Description	Settings
HSSI_SYS_INIT1_WEIGHT 7–4	0	**HSSI Internal Interface 1 Arbitration Weight** Sets the initial weight given to the internal interface 1 (MBus side) of the HSSI. The interface can have a maximum weight of 16.	0000 Weight = 1 0001 Weight = 2 0010 Weight = 3 0011 Weight = 4 0100 Weight = 5 0101 Weight = 6 0110 Weight = 7 0111 Weight = 8 1000 Weight = 9 1001 Weight = 10 1010 Weight = 11 1011 Weight = 12 1100 Weight = 13 1101 Weight = 14 1110 Weight = 15 1111 Weight = 16
HSSI_SYS_INIT0_WEIGHT 3–0	0	**HSSI Internal Interface 0 Arbitration Weight** Sets the initial weight given to the internal interface 0 (MBus side) of the HSSI. The interface can have a maximum weight of 16.	0000 Weight = 1 0001 Weight = 2 0010 Weight = 3 0011 Weight = 4 0100 Weight = 5 0101 Weight = 6 0110 Weight = 7 0111 Weight = 8 1000 Weight = 9 1001 Weight = 10 1010 Weight = 11 1011 Weight = 12 1100 Weight = 13 1101 Weight = 14 1110 Weight = 15 1111 Weight = 16

8.2.29 DMA Request0 Control Register (GCR_DREQ0)

GCR_DREQ0 DMA Request0 Control Register Offset 0x120

Bit	31	30	29	28	27	26	25	24
	DMA_DREQ0_D15	DMA_DREQ0_S15	DMA_DREQ0_D14	DMA_DREQ0_S14	DMA_DREQ0_D13	DMA_DREQ0_S13	DMA_DREQ0_D12	DMA_DREQ0_S12
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMA_DREQ0_D11	DMA_DREQ0_S11	DMA_DREQ0_D10	DMA_DREQ0_S10	DMA_DREQ0_D9	DMA_DREQ0_S9	DMA_DREQ0_D8	DMA_DREQ0_S8
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DMA_DREQ0_D7	DMA_DREQ0_S7	DMA_DREQ0_D6	DMA_DREQ0_S6	DMA_DREQ0_D5	DMA_DREQ0_S5	DMA_DREQ0_D4	DMA_DREQ0_S4
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DMA_DREQ0_D3	DMA_DREQ0_S3	DMA_DREQ0_D2	DMA_DREQ0_S2	DMA_DREQ0_D1	DMA_DREQ0_S1	DMA_DREQ0_D0	DMA_DREQ0_S0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR_DREQ0 associates an external peripheral DMA request 0 to the DMA Controller channel with its destination or source. The user must clear the corresponding bits for a channel for memory-only transactions.

Table 8-29. GCR_DREQ0 Bit Descriptions

Name	Reset	Description	Settings
DMA_DREQ0_D15 31	0	**DMA DREQ0 Destination Channel 15** Associates the DREQ with the destination for Channel 15.	0 DREQ not associated with Channel 15 destination. 1 DREQ associated with Channel 15 destination.
DMA_DREQ0_S15 30	0	**DMA DREQ0 Source Channel 15** Associates the DREQ with the source for Channel 15.	0 DREQ not associated with Channel 15 source. 1 DREQ associated with Channel 15 source.
DMA_DREQ0_D14 29	0	**DMA DREQ0 Destination Channel 14** Associates the DREQ with the destination for Channel 14.	0 DREQ not associated with Channel 14 destination. 1 DREQ associated with Channel 14 destination.

Table 8-29. GCR_DREQ0 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA_DREQ0_S14 28	0	**DMA DREQ0 Source Channel 14** Associates the DREQ with the source for Channel 14.	0 DREQ not associated with Channel 14 source. 1 DREQ associated with Channel 14 source.
DMA_DREQ0_D13 27	0	**DMA DREQ0 Destination Channel 13** Associates the DREQ with the destination for Channel 13.	0 DREQ not associated with Channel 13 destination. 1 DREQ associated with Channel 13 destination.
DMA_DREQ0_S13 26	0	**DMA DREQ0 Source Channel 13** Associates the DREQ with the source for Channel 13.	0 DREQ not associated with Channel 13 source. 1 DREQ associated with Channel 13 source.
DMA_DREQ0_D12 25	0	**DMA DREQ0 Destination Channel 12** Associates the DREQ with the destination for Channel 12.	0 DREQ not associated with Channel 12 destination. 1 DREQ associated with Channel 12 destination.
DMA_DREQ0_S12 24	0	**DMA DREQ0 Source Channel 12** Associates the DREQ with the source for Channel 12.	0 DREQ not associated with Channel 12 source. 1 DREQ associated with Channel 12 source.
DMA_DREQ0_D11 23	0	**DMA DREQ0 Destination Channel 11** Associates the DREQ with the destination for Channel 11.	0 DREQ not associated with Channel 11 destination. 1 DREQ associated with Channel 11 destination.
DMA_DREQ0_S11 22	0	**DMA DREQ0 Source Channel 11** Associates the DREQ with the source for Channel 11.	0 DREQ not associated with Channel 11 source. 1 DREQ associated with Channel 11 source.
DMA_DREQ0_D10 21	0	**DMA DREQ0 Destination Channel 10** Associates the DREQ with the destination for Channel 10.	0 DREQ not associated with Channel 10 destination. 1 DREQ associated with Channel 10 destination.
DMA_DREQ0_S10 20	0	**DMA DREQ0 Source Channel 10** Associates the DREQ with the source for Channel 10.	0 DREQ not associated with Channel 10 source. 1 DREQ associated with Channel 10 source.
DMA_DREQ0_D9 19	0	**DMA DREQ0 Destination Channel 9** Associates the DREQ with the destination for Channel 9.	0 DREQ not associated with Channel 9 destination. 1 DREQ associated with Channel 9 destination.
DMA_DREQ0_S9 18	0	**DMA DREQ0 Source Channel 9** Associates the DREQ with the source for Channel 9.	0 DREQ not associated with Channel 9 source. 1 DREQ associated with Channel 9 source.

Table 8-29. GCR_DREQ0 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA_DREQ0_D8 17	0	**DMA DREQ0 Destination Channel 8** Associates the DREQ with the destination for Channel 8.	0 DREQ not associated with Channel 8 destination. 1 DREQ associated with Channel 8 destination.
DMA_DREQ0_S8 16	0	**DMA DREQ0 Source Channel 8** Associates the DREQ with the source for Channel 8.	0 DREQ not associated with Channel 8 source. 1 DREQ associated with Channel 8 source.
DMA_DREQ0_D7 15	0	**DMA DREQ0 Destination Channel 7** Associates the DREQ with the destination for Channel 7.	0 DREQ not associated with Channel 7 destination. 1 DREQ associated with Channel 7 destination.
DMA_DREQ0_S7 14	0	**DMA DREQ0 Source Channel 7** Associates the DREQ with the source for Channel 7.	0 DREQ not associated with Channel 7 source. 1 DREQ associated with Channel 7 source.
DMA_DREQ0_D6 13	0	**DMA DREQ0 Destination Channel 6** Associates the DREQ with the destination for Channel 6.	0 DREQ not associated with Channel 6 destination. 1 DREQ associated with Channel 6 destination.
DMA_DREQ0_S6 12	0	**DMA DREQ0 Source Channel 6** Associates the DREQ with the source for Channel 6.	0 DREQ not associated with Channel 6 source. 1 DREQ associated with Channel 6 source.
DMA_DREQ0_D5 11	0	**DMA DREQ0 Destination Channel 5** Associates the DREQ with the destination for Channel 5.	0 DREQ not associated with Channel 5 destination. 1 DREQ associated with Channel 5 destination.
DMA_DREQ0_S5 10	0	**DMA DREQ0 Source Channel 5** Associates the DREQ with the source for Channel 5.	0 DREQ not associated with Channel 5 source. 1 DREQ associated with Channel 5 source.
DMA_DREQ0_D4 9	0	**DMA DREQ0 Destination Channel 4** Associates the DREQ with the destination for Channel 4.	0 DREQ not associated with Channel 4 destination. 1 DREQ associated with Channel 4 destination.
DMA_DREQ0_S4 8	0	**DMA DREQ0 Source Channel 4** Associates the DREQ with the source for Channel 4.	0 DREQ not associated with Channel 4 source. 1 DREQ associated with Channel 4 source.
DMA_DREQ0_D3 7	0	**DMA DREQ0 Destination Channel 3** Associates the DREQ with the destination for Channel 3.	0 DREQ not associated with Channel 3 destination. 1 DREQ associated with Channel 3 destination.

Table 8-29. GCR_DREQ0 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA_DREQ0_S3 6	0	**DMA DREQ0 Source Channel 3** Associates the DREQ with the source for Channel 3.	0 DREQ not associated with Channel 3 source. 1 DREQ associated with Channel 3 source.
DMA_DREQ0_D2 5	0	**DMA DREQ0 Destination Channel 2** Associates the DREQ with the destination for Channel 2.	0 DREQ not associated with Channel 2 destination. 1 DREQ associated with Channel 2 destination.
DMA_DREQ0_S2 4	0	**DMA DREQ0 Source Channel 2** Associates the DREQ with the source for Channel 2.	0 DREQ not associated with Channel 2 source. 1 DREQ associated with Channel 2 source.
DMA_DREQ0_D1 3	0	**DMA DREQ0 Destination Channel 1** Associates the DREQ with the destination for Channel 1.	0 DREQ not associated with Channel 1 destination. 1 DREQ associated with Channel 1 destination.
DMA_DREQ0_S1 2	0	**DMA DREQ0 Source Channel 1** Associates the DREQ with the source for Channel 1.	0 DREQ not associated with Channel 1 source. 1 DREQ associated with Channel 1 source.
DMA_DREQ0_D0 1	0	**DMA DREQ0 Destination Channel 0** Associates the DREQ with the destination for Channel 0.	0 DREQ not associated with Channel 0 destination. 1 DREQ associated with Channel 0 destination.
DMA_DREQ0_S0 0	0	**DMA DREQ0 Source Channel 0** Associates the DREQ with the source for Channel 0.	0 DREQ not associated with Channel 0 source. 1 DREQ associated with Channel 0 source.

8.2.30 DMA Request1 Control Register (GCR_DREQ1)

GCR_DREQ1 DMA Request1 Control Register Offset 0x124

Bit	31	30	29	28	27	26	25	24
	DMA_DREQ1_D15	DMA_DREQ1_S15	DMA_DREQ1_D14	DMA_DREQ1_S14	DMA_DREQ1_D13	DMA_DREQ1_S13	DMA_DREQ1_D12	DMA_DREQ1_S12
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMA_DREQ1_D11	DMA_DREQ1_S11	DMA_DREQ1_D10	DMA_DREQ1_S10	DMA_DREQ1_D9	DMA_DREQ1_S9	DMA_DREQ1_D8	DMA_DREQ1_S8
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DMA_DREQ1_D7	DMA_DREQ1_S7	DMA_DREQ1_D6	DMA_DREQ1_S6	DMA_DREQ1_D5	DMA_DREQ1_S5	DMA_DREQ1_D4	DMA_DREQ1_S4
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DMA_DREQ1_D3	DMA_DREQ1_S3	DMA_DREQ1_D2	DMA_DREQ1_S2	DMA_DREQ1_D1	DMA_DREQ1_S1	DMA_DREQ1_D0	DMA_DREQ1_S0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR_DREQ1 associates an external peripheral DMA request 1 to the DMA Controller channel with its destination or source. The user must clear the corresponding bits for a channel for memory-only transactions.

Table 8-30. GCR_DREQ1 Bit Descriptions

Name	Reset	Description	Settings
DMA_DREQ1_D15 31	0	**DMA DREQ1 Destination Channel 15** Associates the DREQ with the destination for Channel 15.	0 DREQ not associated with Channel 15 destination. 1 DREQ associated with Channel 15 destination.
DMA_DREQ1_S15 30	0	**DMA DREQ1 Source Channel 15** Associates the DREQ with the source for Channel 15.	0 DREQ not associated with Channel 15 source. 1 DREQ associated with Channel 15 source.
DMA_DREQ1_D14 29	0	**DMA DREQ1 Destination Channel 14** Associates the DREQ with the destination for Channel 14.	0 DREQ not associated with Channel 14 destination. 1 DREQ associated with Channel 14 destination.

Table 8-30. GCR_DREQ1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA_DREQ1_S14 28	0	**DMA DREQ1 Source Channel 14** Associates the DREQ with the source for Channel 14.	0 DREQ not associated with Channel 14 source. 1 DREQ associated with Channel 14 source.
DMA_DREQ1_D13 27	0	**DMA DREQ1 Destination Channel 13** Associates the DREQ with the destination for Channel 13.	0 DREQ not associated with Channel 13 destination. 1 DREQ associated with Channel 13 destination.
DMA_DREQ1_S13 26	0	**DMA DREQ1 Source Channel 13** Associates the DREQ with the source for Channel 13.	0 DREQ not associated with Channel 13 source. 1 DREQ associated with Channel 13 source.
DMA_DREQ1_D12 25	0	**DMA DREQ1 Destination Channel 12** Associates the DREQ with the destination for Channel 12.	0 DREQ not associated with Channel 12 destination. 1 DREQ associated with Channel 12 destination.
DMA_DREQ1_S12 24	0	**DMA DREQ1 Source Channel 12** Associates the DREQ with the source for Channel 12.	0 DREQ not associated with Channel 12 source. 1 DREQ associated with Channel 12 source.
DMA_DREQ1_D11 23	0	**DMA DREQ1 Destination Channel 11** Associates the DREQ with the destination for Channel 11.	0 DREQ not associated with Channel 11 destination. 1 DREQ associated with Channel 11 destination.
DMA_DREQ1_S11 22	0	**DMA DREQ1 Source Channel 11** Associates the DREQ with the source for Channel 11.	0 DREQ not associated with Channel 11 source. 1 DREQ associated with Channel 11 source.
DMA_DREQ1_D10 21	0	**DMA DREQ1 Destination Channel 10** Associates the DREQ with the destination for Channel 10.	0 DREQ not associated with Channel 10 destination. 1 DREQ associated with Channel 10 destination.
DMA_DREQ1_S10 20	0	**DMA DREQ1 Source Channel 10** Associates the DREQ with the source for Channel 10.	0 DREQ not associated with Channel 10 source. 1 DREQ associated with Channel 10 source.
DMA_DREQ1_D9 19	0	**DMA DREQ1 Destination Channel 9** Associates the DREQ with the destination for Channel 9.	0 DREQ not associated with Channel 9 destination. 1 DREQ associated with Channel 9 destination.
DMA_DREQ1_S9 18	0	**DMA DREQ1 Source Channel 9** Associates the DREQ with the source for Channel 9.	0 DREQ not associated with Channel 9 source. 1 DREQ associated with Channel 9 source.

Table 8-30. GCR_DREQ1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA_DREQ1_D8 17	0	**DMA DREQ1 Destination Channel 8** Associates the DREQ with the destination for Channel 8.	0 DREQ not associated with Channel 8 destination. 1 DREQ associated with Channel 8 destination.
DMA_DREQ1_S8 16	0	**DMA DREQ1 Source Channel 8** Associates the DREQ with the source for Channel 8.	0 DREQ not associated with Channel 8 source. 1 DREQ associated with Channel 8 source.
DMA_DREQ1_D7 15	0	**DMA DREQ1 Destination Channel 7** Associates the DREQ with the destination for Channel 7.	0 DREQ not associated with Channel 7 destination. 1 DREQ associated with Channel 7 destination.
DMA_DREQ1_S7 14	0	**DMA DREQ1 Source Channel 7** Associates the DREQ with the source for Channel 7.	0 DREQ not associated with Channel 7 source. 1 DREQ associated with Channel 7 source.
DMA_DREQ1_D6 13	0	**DMA DREQ1 Destination Channel 6** Associates the DREQ with the destination for Channel 6.	0 DREQ not associated with Channel 6 destination. 1 DREQ associated with Channel 6 destination.
DMA_DREQ1_S6 12	0	**DMA DREQ1 Source Channel 6** Associates the DREQ with the source for Channel 6.	0 DREQ not associated with Channel 6 source. 1 DREQ associated with Channel 6 source.
DMA_DREQ1_D5 11	0	**DMA DREQ1 Destination Channel 5** Associates the DREQ with the destination for Channel 5.	0 DREQ not associated with Channel 5 destination. 1 DREQ associated with Channel 5 destination.
DMA_DREQ1_S5 10	0	**DMA DREQ1 Source Channel 5** Associates the DREQ with the source for Channel 5.	0 DREQ not associated with Channel 5 source. 1 DREQ associated with Channel 5 source.
DMA_DREQ1_D4 9	0	**DMA DREQ1 Destination Channel 4** Associates the DREQ with the destination for Channel 4.	0 DREQ not associated with Channel 4 destination. 1 DREQ associated with Channel 4 destination.
DMA_DREQ1_S4 8	0	**DMA DREQ1 Source Channel 4** Associates the DREQ with the source for Channel 4.	0 DREQ not associated with Channel 4 source. 1 DREQ associated with Channel 4 source.
DMA_DREQ1_D3 7	0	**DMA DREQ1 Destination Channel 3** Associates the DREQ with the destination for Channel 3.	0 DREQ not associated with Channel 3 destination. 1 DREQ associated with Channel 3 destination.

Table 8-30. GCR_DREQ1 Bit Descriptions (Continued)

Name	Reset	Description	Settings
DMA_DREQ1_S3 6	0	**DMA DREQ1 Source Channel 3** Associates the DREQ with the source for Channel 3.	0 DREQ not associated with Channel 3 source. 1 DREQ associated with Channel 3 source.
DMA_DREQ1_D2 5	0	**DMA DREQ1 Destination Channel 2** Associates the DREQ with the destination for Channel 2.	0 DREQ not associated with Channel 2 destination. 1 DREQ associated with Channel 2 destination.
DMA_DREQ1_S2 4	0	**DMA DREQ1 Source Channel 2** Associates the DREQ with the source for Channel 2.	0 DREQ not associated with Channel 2 source. 1 DREQ associated with Channel 2 source.
DMA_DREQ1_D1 3	0	**DMA DREQ1 Destination Channel 1** Associates the DREQ with the destination for Channel 1.	0 DREQ not associated with Channel 1 destination. 1 DREQ associated with Channel 1 destination.
DMA_DREQ1_S1 2	0	**DMA DREQ1 Source Channel 1** Associates the DREQ with the source for Channel 1.	0 DREQ not associated with Channel 1 source. 1 DREQ associated with Channel 1 source.
DMA_DREQ1_D0 1	0	**DMA DREQ1 Destination Channel 0** Associates the DREQ with the destination for Channel 0.	0 DREQ not associated with Channel 0 destination. 1 DREQ associated with Channel 0 destination.
DMA_DREQ1_S0 0	0	**DMA DREQ1 Source Channel 0** Associates the DREQ with the source for Channel 0.	0 DREQ not associated with Channel 0 source. 1 DREQ associated with Channel 0 source.

8.2.31 DMA Done Control Register (GCR_DDONE)

GCR_DDONE DMA Done Control Register Offset 0x128

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—		V1	DONE1_CH				
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		V0	DONE0_CH				
Type	R/W							
Reset	0	0	0	0	0	0	0	0

GCR_DONE controls DMA external request DONE for requestor 1 and 2.

Table 8-31. GCR_DDONE Bit Descriptions

Name	Reset	Description	Settings
— 31–14	0	Reserved. Write to zero for future compatibility.	
V1 13	0	**Valid DONE1** Indicates whether the value of DONE1_CH is valid.	0 DONE1_CH is not valid. 1 DONE1_CH is valid.

Table 8-31. GCR_DDONE Bit Descriptions (Continued)

Name	Reset	Description	Settings
DONE1_CH 12–8	0	**DMA DONE1 Channel Select** Defines the unidirectional channel that drives the DONE1 signal. The signal is valid when GCR_DDONE[V1] is set.	00000 Channel 0 source. 00001 Channel 0 destination. 00010 Channel 1 source. 00011 Channel 1 destination. 00100 Channel 2 source. 00101 Channel 2 destination. 00110 Channel 3 source. 00111 Channel 3 destination. 01000 Channel 4 source. 01001 Channel 4 destination. 01010 Channel 5 source. 01011 Channel 5 destination. 01100 Channel 6 source. 01101 Channel 6 destination. 01110 Channel 7 source. 01111 Channel 7 destination. 10000 Channel 8 source. 10001 Channel 8 destination. 10010 Channel 9 source. 10011 Channel 9 destination. 10100 Channel 10 source. 10101 Channel 10 destination. 10110 Channel 11 source. 10111 Channel 11 destination. 11000 Channel 12 source. 11001 Channel 12 destination. 11010 Channel 13 source. 11011 Channel 13 destination. 11100 Channel 14 source. 11101 Channel 14 destination. 11110 Channel 15 source. 11111 Channel 15 destination.
— 7–6	0	Reserved. Write to zero for future compatibility.	

Table 8-31. GCR_DDONE Bit Descriptions (Continued)

Name	Reset	Description	Settings
V0 5	0	**Valid DONE0** Indicates whether the value of DONE0_CH is valid.	0 DONE0_CH is not valid. 1 DONE0_CH is valid.
DONE0_CH 4–0	0	**DMA DONE0 Channel Select** Defines the unidirectional channel that drives the DONE0 signal. The signal is valid when GCR_DDONE[V0] is set.	00000 Channel 0 source. 00001 Channel 0 destination. 00010 Channel 1 source. 00011 Channel 1 destination. 00100 Channel 2 source. 00101 Channel 2 destination. 00110 Channel 3 source. 00111 Channel 3 destination. 01000 Channel 4 source. 01001 Channel 4 destination. 01010 Channel 5 source. 01011 Channel 5 destination. 01100 Channel 6 source. 01101 Channel 6 destination. 01110 Channel 7 source. 01111 Channel 7 destination. 10000 Channel 8 source. 10001 Channel 8 destination. 10010 Channel 9 source. 10011 Channel 9 destination. 10100 Channel 10 source. 10101 Channel 10 destination. 10110 Channel 11 source. 10111 Channel 11 destination. 11000 Channel 12 source. 11001 Channel 12 destination. 11010 Channel 13 source. 11011 Channel 13 destination. 11100 Channel 14 source. 11101 Channel 14 destination. 11110 Channel 15 source. 11111 Channel 15 destination.

8.2.32 DDR1 General Configuration Register (DDR1_GCR)

DDR1_GCR DDR1 General Configuration Register Offset 0x12C

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—					DDR1_STOP	DDR1_DOZE	DDR1_PWR_DOWN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

DDR1_GCR controls part of the DDR1 operation. All bits are cleared on a hard reset event.

Table 8-32. DDR1_GCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
DDR1_STOP 2	0	**DDR1 Stop** With the DDRC Stop ACK, actively stops the controller and DDR memory clock. If the memory is not put in self-refresh mode prior to activating Stop mode, data is lost.	0 DDR1 Stop not requested. 1 DDR1 Stop requested.
DDR1_DOZE 1	0	**DDR1 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the DDR controller 1 is stopped. **Note:** To conserve power, when DDR1 is not used by an application, set this bit as soon as possible after reset. This step should be included as part of the initialization code.	0 Normal operation. 1 Doze mode.
DDR1_POWER_DOWN 0	0	**DDR1 Power Down** When set, powers down all DDR1 areas with gated clocks.	0 DDR1 power up. 1 DDR1 power down.

8.2.33 DDR2 General Configuration Register (DDR2_GCR)

DDR2_GCR DDR2 General Configuration Register Offset 0x130

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—					DDR2_STOP	DDR2_DOZE	DDR2_PWR_DOWN
Type	R/W							
Reset	0	0	0	0	0	0	0	0

DDR2_GCR controls part of the DDR2 operation. All bits are cleared on a hard reset event.

Table 8-33. DDR2_GCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
DDR2_STOP 2	0	**DDR2 Stop** With the DDRC Stop ACK, actively stops the controller and DDR memory clock. If the memory is not put in self-refresh mode prior to activating Stop mode, data is lost.	0 DDR2 Stop not requested. 1 DDR2 Stop requested.
DDR2_DOZE 1	0	**DDR2 Doze** Used to select Doze mode, in which all register accesses are acknowledged, but writes are not written and reads do not contain valid data. Setting this bit prevents lockup of the device internal bus while the DDR2 controller is stopped. **Note:** To conserve power, when DDR2 is not used by an application, set this bit as soon as possible after reset. This step should be included as part of the initialization code.	0 Normal operation. 1 Doze mode.
DDR2_POWER_DOWN 0	0	**DDR2 Power Down** When set, powers down all DDR2 areas with gated clocks.	0 DDR2 power up. 1 DDR2 power down.

8.2.34 Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR)

CORE_SLV_GCR Core Subsystem Slave Port General Configuration Register Offset 0x138

Bit	31	30	29	28	27	26	25	24
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	—							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	—		CORE_SLV_WIN_CLOSE5	CORE_SLV_WIN_CLOSE4	CORE_SLV_WIN_CLOSE3	CORE_SLV_WIN_CLOSE2	CORE_SLV_WIN_CLOSE1	CORE_SLV_WIN_CLOSE0
Type	R/W							
Reset	0	0	0	0	0	0	0	0

CORE_SLV_GCR controls the status of the slave ports for the core subsystems.

Table 8-34. CORE_SLV_GCR Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to zero for future compatibility.	
CORE_SLV_WIN_CLOSE5 5	0	**Peripheral Access to Core 5 L2/M2 Disable** Set the bit to 1 to disable accesses by the device peripherals (that is, DMAC, QUICC Engine module, and so on) to M2/L2 in core 5.	0 Enable peripheral access to Core 5 M2/L2 memory. 1 Disable peripheral access to Core 5 M2/L2 memory.
CORE_SLV_WIN_CLOSE4 4	0	**Peripheral Access to Core 4 L2/M2 Disable** Set the bit to 1 to disable accesses by the device peripherals (that is, DMAC, QUICC Engine module, and so on) to M2/L2 in core 4.	0 Enable peripheral access to Core 4 M2/L2 memory. 1 Disable peripheral access to Core 4 M2/L2 memory.
CORE_SLV_WIN_CLOSE3 3	0	**Peripheral Access to Core 3 L2/M2 Disable** Set the bit to 1 to disable accesses by the device peripherals (that is, DMAC, QUICC Engine module, and so on) to M2/L2 in core 3.	0 Enable peripheral access to Core 3 M2/L2 memory. 1 Disable peripheral access to Core 3 M2/L2 memory.
CORE_SLV_WIN_CLOSE2 2	0	**Peripheral Access to Core 2 L2/M2 Disable** Set the bit to 1 to disable accesses by the device peripherals (that is, DMAC, QUICC Engine module, and so on) to M2/L2 in core 2.	0 Enable peripheral access to Core 2 M2/L2 memory. 1 Disable peripheral access to Core 2 M2/L2 memory.

Table 8-34. CORE_SLV_GCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
CORE_SLV_WIN_CLOSE1 1	0	**Peripheral Access to Core 1 L2/M2 Disable** Set the bit to 1 to disable accesses by the device peripherals (that is, DMAC, QUICC Engine module, and so on) to M2/L2 in core 1.	0 Enable peripheral access to Core 15 M2/L2 memory. 1 Disable peripheral access to Core 1 M2/L2 memory.
CORE_SLV_WIN_CLOSE0 0	0	**Peripheral Access to Core 0 L2/M2 Disable** Set the bit to 1 to disable accesses by the device peripherals (that is, DMAC, QUICC Engine module, and so on) to M2/L2 in core 0.	0 Enable peripheral access to Core 0 M2/L2 memory. 1 Disable peripheral access to Core 0 M2/L2 memory.

Memory Map 9

This section describes the memory map of MSC8156E.

The MSC8156E incorporates four address groups:

- Shared memory (M3, DDR, QUICC Engine subsystem, and MAPLE-B) address space.
- Shared SC3850 DSP core subsystem M2/L2 memories
- SC3850 DSP core subsystem internal address space is accessible only by the SC3850 core. Each SC3850 core can access its own internal address space.
- Control Configuration and Status Registers (CCSR) address space.

9.1 Shared Memory Address Space

The shared memory address space resides within addresses 0x40000000–0xFEFFFFFF. It includes the M3 memory, two DDR memories, MAPLE-B, QUICC Engine subsystem, and the boot ROM.

Note: The CCSR address space is not part of the shared memory space because the SC3850 DSP core subsystem internal memory resides in the middle (the CCSR address space starts at address 0xFFF10000).

Table 9-1. Shared Memory Address Space

Address	Purpose	Size in Bytes
0x40000000–0x5FFFFFFF	DDR1 Memory (default value)	512 M
0x60000000–0x7FFFFFFF	Reserved; used for DDR1 memory if configured for 1 GB.	512 M
0x80000000–0x9FFFFFFF	DDR2 Memory (default value)	512 M
0xA0000000–0xBFFFFFFF	Reserved; used for DDR2 memory if configured for 1 GB	512 M
0xC0000000–0xC0107FFF	M3 Memory	1 M + 32 K
0xC0108000–0xFECFFFFF	Reserved	511 M – 32 K
0xFED00000–0xFEDFFFFF	MAPLE-B	1 M
0xFEE00000–0xFEE3FFFF	QUICC Engine Subsystem	256 K
0xFEE40000–0xFEEFFFFF	Reserved (QUICC Engine Subsystem)	768 K
0xFEF00000–0xFEF17FFF	Boot ROM	96 K
0xFEF18000–0xFEFFFFFF	Reserved	928 K

9.2 Shared SC3850 DSP Core Subsystem M2/L2 Memories

Each SC3850 core subsystem includes 512 KB of level-2 memory that can be partitioned into M2 memory and L2 cache. The minimal size of each partition is 64 KB and each core can have a unique combination of M2 and L2 memory partitions. The partition size can be changed during operation, but the user must make sure that all ongoing accesses are terminated before making the change. The default partitioning out of reset defines all of the level-2 memory as shared M2 memory. The boot program configures the MMU to support that configuration and sets the CLASS values accordingly.

When partitioned as M2 memory, the memory is available as a shared memory that can be used as private memory by the core to which it belongs and accessed by all other system masters and external hosts, but not by the other cores. Although the M2 memories have different physical addresses, all cores use a virtual address of 0x0 to access its own M2 memory and uses MMU translation to access the correct memory space.

When partitioned as L2 cache memory, the L2 cache can be accessed by the core to which it belongs and by other masters in the system, but not by other cores. The L2 accesses are translated through the core subsystem slave ports to the L2 cache, allowing DDR access for specific core cache. The size in bytes shown in **Table 9-2** indicates the total DDR access range.

The shared M2/L2 memory space resides within the address range 0x20000000–0x3FFFFFFF and includes all cores.

As shown at the **Table 9-2**, when an initiator other than the core accesses L2 cache, its address range is limited to 32 MB. Before hitting the cache, this address range is always mapped to the first 32 MB of DDR0 which uses the address range 0x40000000–0x41FFFFFF. The mapping is the same for all DSP subsystems. If L2 acts as a shared cache by the core and the additional initiators, this feature has the following benefits:

- In the DMA model in which the DMA controller transfers data to L2 cache and the core can use it, the scheduling restriction between DMA job completion and core job start is relaxed (in comparison to DMA and M2). If the core starts reading data not written by the DMA controller, a miss is generated to the DDR and data is read directly.
- If an initiator must read the same data from DDR several times to save DDR bandwidth, the data resides in L2 after the first read and is than read from L2 repeatedly without having to access the DDR memory.
- This feature can be used in conjunction with cache partitioning in which the cache can be used as two separate caches, one for the core and the other for another initiator.

Table 9-2. Shared M2/L2 Memories Address Space

Address	Purpose	Size in Bytes
0x20000000–0x23FFFFFF	Reserved	64 M
0x24000000–0x25FFFFFF	DDR through Core 5 L2	32 M

Table 9-2. Shared M2/L2 Memories Address Space (Continued)

Address	Purpose	Size in Bytes
0x26000000–0x27FFFFFF	DDR through Core 4 L2	32 M
0x28000000–0x29FFFFFF	DDR through Core 3 L2	32 M
0x2A000000–0x2BFFFFFF	DDR through Core 2 L2	32 M
0x2C000000–0x2DFFFFFF	DDR through Core 1 L2	32 M
0x2E000000–0x2FFFFFFF	DDR through Core 0 L2	32 M
0x30000000–0x3007FFFF	Core 0 M2 memory	512 K max.
0x30080000–0x30FFFFFF	Reserved	16 M – 512 K
0x31000000–0x3107FFFF	Core 1 M2 memory	512 K max.
0x31080000–0x31FFFFFF	Reserved	16 M – 512 K
0x32000000–0x3207FFFF	Core 2 M2 memory	512 K max.
0x32080000–0x32FFFFFF	Reserved	16 M – 512 K
0x33000000–0x3307FFFF	Core 3 M2 memory	512 K max.
0x33080000–0x33FFFFFF	Reserved	16 M – 512 K
0x34000000–0x3407FFFF	Core 4 M2 memory	512 K max.
0x34080000–0x34FFFFFF	Reserved	16 M – 512 K
0x35000000–0x3507FFFF	Core 5 M2 memory	512 K max.
0x35080000–0x37FFFFFF	Reserved	48 M – 512 K
Note: Each of the M2 memory areas is configurable in size in 64 KB increments starting from the lowest 64 KB address block up to the maximum 512 KB. Any blocks not allocated as M2 memory are reserved.		

9.3 SC3850 DSP Core Subsystem Internal Address Space

Each SC3850 core can access the internal address space of its DSP core subsystem. The SC3850 internal address space is located from address 0xFF000000–0xFFF0FFFF (15 MB + 64 KB). **Table 9-3** lists details for the DSP core subsystem internal address space.

Table 9-3. SC3850 DSP Core Subsystem Internal Address Space

Address	Purpose	Size (Bytes)	Remarks
0xFF000000–0xFFEFFDFF	Reserved	15 M – 512	
0xFFEFFE00–0xFFEFFFFF	OCE	512	User/Supervisor
0xFFF00000[2]–0xFFF003FF	Reserved	1K	
0xFFF00400–0xFFF007FF	EPIC	1K	Supervisor
0xFFF00800–0xFFF00BFF	Data Cache registers	1K	Supervisor
0xFFF00C00–0xFFF00FFF	Instruction Cache registers	1K	Supervisor
0xFFF01000–0xFFF05FFF	Reserved	20 K	
0xFFF06000–0xFFF09FFF	MMU	16 K	Supervisor
0xFFF0A000–0xFFF0A2FF	DPU	768	User/Supervisor
0xFFF0A300–0xFFF0A3FF	TIMERS	256	User/Supervisor
0xFFF0A400–0xFFF0FFFF	Reserved	23 K	
Note: Access in both User and Supervisor modes is allowed only if enabled via the EMR[EMEA] bit in the core.			

Note: See the *MSC8156 SC3850 DSP Subsystem Reference Manual* for details.

9.4 CCSR Address Space

The MSC8156E CCSR is mapped within a contiguous block of memory. The size of the CCSR in MSC8156E is 956 KB. **Table 9-4** details the CCSR address space.

Table 9-4. CCSR Address Space

Address	Purpose	Size (Bytes)
0xFFF10000–0xFFF103FF	DMA	1 K
0xFFF10400–0xFFF17FFF	Reserved	31 K
0xFFF18000–0xFFF18FFF	CLASS Registers	4 K
0xFFF19000–0xFFF1BFFF	Reserved	12 K
0xFFF1C000–0xFFF1FFFF	MAPLE-B Registers	16 K
0xFFF20000–0xFFF21FFF	DDR Controller 1	8 K
0xFFF22000–0xFFF23FFF	DDR Controller 2	8 K
0xFFF24000–0xFFF2407F	Clock	128
0xFFF24080–0xFFF247FF	reserved	2 K – 128
0xFFF24800–0xFFF248FF	Reset	256
0xFFF24900–0xFFF24BFF	reserved	768
0xFFF24C00–0xFFF24CFF	I^2C	256
0xFFF24D00–0xFFF24FFF	reserved	768
0xFFF25000–0xFFF250FF	Watch Dog Timers	256
0xFFF25100–0xFFF251FF		256
0xFFF25200–0xFFF252FF		256
0xFFF25300–0xFFF253FF		256
0xFFF25400–0xFFF254FF		256
0xFFF25500–0xFFF255FF		256
0xFFF25600–0xFFF256FF		256
0xFFF25700–0xFFF257FF		256
0xFFF25800–0xFFF25FFF	reserved	2K
0xFFF26000–0xFFF260FF	Timers	256
0xFFF26100–0xFFF261FF		256
0xFFF26200–0xFFF262FF		256
0xFFF26300–0xFFF263FF		256
0xFFF26400–0xFFF26BFF	Reserved	2K
0xFFF26C00–0xFFF26C3F	UART	64
0xFFF26C40–0xFFF26FFF	Reserved	1K – 64
0xFFF27000–0xFFF270FF	GIC Registers	256
0xFFF27100–0xFFF271FF	Hardware Semaphores	256
0xFFF27200–0xFFF272FF	GPIO Registers	256
0xFFF27300–0xFFF27FFF	Reserved	4 K – 768
0xFFF28000–0xFFF281FF	General Configuration Registers	512
0xFFF28200–0xFFF2FFFF	Reserved	16 K – 32
0xFFF30000–0xFFF33FFF	TDM0 Registers	16 K
0xFFF34000–0xFFF37FFF	TDM1 Registers	16 K
0xFFF38000–0xFFF3BFFF	TDM2 Registers	16 K
0xFFF3C000–0xFFF3FFFF	TDM3 Registers	16 K
0xFFF40000–0xFFF7FFFF	Reserved	256 K
0xFFF80000–0xFFF9FFFF	RapidIO Registers	128 K
0xFFFA0000–0xFFFA0FFF	Reserved	4 K
0xFFFA1000–0xFFFA103F	OCN Crossbar Switch to MBus0	64

Table 9-4. CCSR Address Space (Continued)

Address	Purpose	Size (Bytes)
0xFFFA1040–0xFFFA107F	OCN Crossbar Switch to MBus1	64
0xFFFA1080–0xFFFA7FFF	Reserved	28 K – 128
0xFFFA8000–0xFFFA8FFF	Dedicated DMA Controller 0 Registers	4 K
0xFFFA9000–0xFFFA9FFF	DMA Controller 0 to OCN	4 K
0xFFFAA000–0xFFFAAFFF	Dedicated DMA Controller 1 Registers	4 K
0xFFFAB000–0xFFFABFFF	DMA Controller 1 to OCN	4 K
0xFFFAC000–0xFFFAC07F	SerDes PHY 0 Registers	128
0xFFFAC080–0xFFFACFFF	Reserved	4 K – 128
0xFFFAD000–0xFFFAD07F	SerDes PHY 1 Registers	128
0xFFFAD080–0xFFFB6FFF	Reserved	40 K – 128
0xFFFB7000–0xFFFB7FFF	PCI Express Registers	4 K
0xFFFB8000–0xFFFBAFFF	Reserved	12 K
0xFFFBB000–0xFFFBB7FF	RapidIO/PCI Express Security Bridge Registers	2 K
0xFFFBB800–0xFFFBB8FF	Performance Monitor Registers	256
0xFFFBB900–0xFFFCFFFF	Reserved	82 K – 256
0xFFFD0000–0xFFFDFFFF	Security Engine Registers	64 K
0xFFFDE000–0xFFFFEFFF	Reserved	128 K

9.5 Initiators Views of the System Address Space

Addressing within the system address space depends on the device addressing the system memory space. The following sections define how the cores, system peripherals, and the Security Engine address this space.

9.5.1 SC3850 (Data) View of the System Address Space

Table 9-5 describes the system address space as seen by each SC3850 core subsystem.

Table 9-5. SC3850 (Data) View of the System Address Space

Address		Purpose	Size (Bytes)
0x00000000–0x2FFFFFFF		Reserved	768 M
The allocation of the memory space from 0x30000000–0x3FFFFFFF is core subsystem dependent as follows:			
Core 0	0x30000000–0x3007FFFF	Private M2 memory (size is configurable)	512 K maximum
	0x30080000–0x3FFFFFFF	Reserved	256 M – 512 K
Core 1	0x30000000–0x30FFFFFF	Reserved	16 M
	0x31000000–0x3107FFFF	Private M2 memory (size is configurable)	512 K maximum
	0x31080000–0x3FFFFFFF	Reserved	240 M – 512 K
Core 2	0x30000000–0x31FFFFFF	Reserved	32 M
	0x32000000–0x3207FFFF	Private M2 memory (size is configurable)	512 K maximum
	0x32080000–0x3FFFFFFF	Reserved	224 M – 512 K
Core 3	0x30000000–0x32FFFFFF	Reserved	48 M
	0x33000000–0x3307FFFF	Private M2 memory (size is configurable)	512 K maximum
	0x33080000–0x3FFFFFFF	Reserved	208 M – 512 K
Core 4	0x30000000–0x33FFFFFF	Reserved	64 M
	0x34000000–0x3407FFFF	Private M2 memory (size is configurable)	512 K maximum
	0x34080000–0x3FFFFFFF	Reserved	192 M – 512 K

Table 9-5. SC3850 (Data) View of the System Address Space

Address		Purpose	Size (Bytes)
Core 5	0x30000000–0x34FFFFFF	Reserved	80 M
	0x35000000–0x3507FFFF	Private M2 memory (size is configurable)	512 K maximum
	0x35080000–0x3FFFFFFF	Reserved	176 M – 512 K
Note: Each core subsystem L2 memory can be allocated in 64 KB increments starting from the lowest 64 KB address block up to the maximum 512 KB. Any blocks not allocated as M2 memory are reserved.			
0x40000000–0xFEFFFFFF		Shared Memory Address Space	3 G – 528 M
0xFF000000–0xFFF0FFFF		SC3850 DSP core subsystem Internal Address Space	15 M + 64 K
0xFFF10000–0xFFFFEFFF		CCSR Address Space	956 K
0xFFFFF000–0xFFFFFFFF		Reserved	4 K

9.5.2 Peripherals View of the System Address Space

Table 9-6 describes the system address space as seen by the MSC8156E peripherals (RapidIO and PCI Express controllers, JTAG, QUICC Engine subsystem, TDM, DMA, and MAPLE-B–both MBus interfaces).

Table 9-6. Peripherals View of the System Address Space

Address	Purpose	Size (Bytes)
0x00000000–0x1FFFFFFF	Reserved	512 M
0x20000000–0x3FFFFFFF	Shared L2/M2 Memory Space	512 M
0x40000000–0xFEFFFFFF	Shared Memory Address Space	3 G – 528 M
0xFF000000–0xFFF0FFFF	Reserved	15M + 64K
0xFFF10000–0xFFFFEFFF	CCSR Address Space	956 K
0xFFFFF000–0xFFFFFFFF	Reserved	4 K

An external initiator (to the MSC8156E device) can generate accesses to the system address space using the:

- Test Access Port/JTAG with direct addressing.
- The RapidIO subsystem/PCI Express controller using the RapidIO/PCI Express inbound address translation.

9.5.3 Security Engine View of the System Address Space

Table 9-7. describes the system address space as seen by the Security Engine (SEC).

Table 9-7. SEC View of the System Address Space

Address	Purpose	Size (Bytes)
0x00000000–0x0FFFFFFF	Reserved	256 M
0x10000000–0x1FFFFFFF	Keys (accessible only by the SEC)	256 M
0x20000000–0x3FFFFFFF	Shared L2/M2 memory space	512 M

Table 9-7. SEC View of the System Address Space

Address	Purpose	Size (Bytes)
0x40000000–0xFEFFFFFF	Shared memory address space	3 G – 528 M
0xFF000000–0xFFFFFFFF	Reserved	16 M

9.6 Detailed System Memory Map

Table 9-8. Detailed System Memory Map

Address	Name/Status	Acronym
0x00000000–0x3FFFFFFF	Reserved	
0x40000000–0xFEFFFFFF	Shared memory	
• 0x40000000–0x5FFFFFFF	DDR1 memory	
• 0x60000000–0x7FFFFFFF	reserved	
• 0x80000000–0x9FFFFFFF	DDR2 Memory	
• 0xA0000000–0xBFFFFFFF	reserved	
• 0xC0000000–0xC010FFFF	M3 Memory	
• 0xC0108000–0xFECFFFFF	reserved	
• 0xFED00000–0xFEDFFFFF	MAPLE-B (see **Chapter 26**, *Multi Accelerator Platform Engine, Baseband (MAPLE-B)*)	
– 0xFED00000–0xFED005FF	Parameter RAM	
– 0xFED00000–0xFED00003	reserved	
– 0xFED00004–0xFED00007	MAPLE BD Rings Configuration Parameter	MBDRCP
– 0xFED00008–0xFED0000F	reserved	
– 0xFED00010	MAPLE UCode Version Number	MUCVP
– 0xFED00014–0xFED00017	reserved	
– 0xFED00018	MAPLE Timer Period Parameter	MP_TPP
– 0xFED0001C–0xFED0007F	reserved	
– 0xFED00080–0xFED00083	MAPLE TVPE BD Ring High Priority A 0 Parameter	MTVBRHPA0P
– 0xFED00084–0xFED00087	MAPLE TVPE BD Ring High Priority B 0 Parameter	MTVBRHPB0P
– 0xFED00088–0xFED0008B	MAPLE TVPE BD Ring High Priority A 1 Parameter	MTVBRHPA1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED0008C–0xFED0008F	MAPLE TVPE BD Ring High Priority B 1 Parameter	MTVBRHPB1P
– 0xFED00090–0xFED00093	MAPLE TVPE BD Ring High Priority A 2 Parameter	MTVBRHPA2P
– 0xFED00094–0xFED00097	MAPLE TVPE BD Ring High Priority B 2 Parameter	MTVBRHPB2P
– 0xFED00098–0xFED0009B	MAPLE TVPE BD Ring High Priority A 3 Parameter	MTVBRHPA3P
– 0xFED0009C–0xFED0009F	MAPLE TVPE BD Ring High Priority B 3 Parameter	MTVBRHPB3P
– 0xFED000A0–0xFED000A3	MAPLE TVPE BD Ring High Priority A 4 Parameter	MTVBRHPA4P
– 0xFED000A4–0xFED000A7	MAPLE TVPE BD Ring High Priority B 4 Parameter	MTVBRHPB4P
– 0xFED000A8–0xFED000AB	MAPLE TVPE BD Ring High Priority A 5 Parameter	MTVBRHPA5P
– 0xFED000AC–0xFED000AF	MAPLE TVPE BD Ring High Priority B 5 Parameter	MTVBRHPB5P
– 0xFED000B0–0xFED000B3	MAPLE TVPE BD Ring High Priority A 6 Parameter	MTVBRHPA6P
– 0xFED000B4–0xFED000B7	MAPLE TVPE BD Ring High Priority B 6 Parameter	MTVBRHPB6P
– 0xFED000B8–0xFED000BB	MAPLE TVPE BD Ring High Priority A 7 Parameter	MTVBRHPA7P
– 0xFED000BC–0xFED000BF	MAPLE TVPE BD Ring High Priority B 7 Parameter	MTVBRHPB7P
– 0xFED000C0–0xFED000C3	MAPLE TVPE BD Ring Low Priority A 0 Parameter	MTVBRLPA0P
– 0xFED000C4–0xFED000C7	MAPLE TVPE BD Ring Low Priority B 0 Parameter	MTVBRLPB0P
– 0xFED000C8–0xFED000CB	MAPLE TVPE BD Ring Low Priority A 1 Parameter	MTVBRLPA1P
– 0xFED000CC–0xFED000CF	MAPLE TVPE BD Ring Low Priority B 1 Parameter	MTVBRLPB1P
– 0xFED000D0–0xFED000D3	MAPLE TVPE BD Ring Low Priority A 2 Parameter	MTVBRLPA2P
– 0xFED000D4–0xFED000D7	MAPLE TVPE BD Ring Low Priority B 2 Parameter	MTVBRLPB2P
– 0xFED000D8–0xFED000DB	MAPLE TVPE BD Ring Low Priority A 3 Parameter	MTVBRLPA3P
– 0xFED000DC–0xFED000DF	MAPLE TVPE BD Ring Low Priority B 3 Parameter	MTVBRLPB3P
– 0xFED000E0–0xFED000E3	MAPLE TVPE BD Ring Low Priority A 4 Parameter	MTVBRLPA4P
– 0xFED000E4–0xFED000E7	MAPLE TVPE BD Ring Low Priority B 4 Parameter	MTVBRLPB4P
– 0xFED000E8–0xFED000EB	MAPLE TVPE BD Ring Low Priority A 5 Parameter	MTVBRLPA5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED000EC–0xFED000EF	MAPLE TVPE BD Ring Low Priority B 5 Parameter	MTVBRLPB5P
– 0xFED000F0–0xFED000F3	MAPLE TVPE BD Ring Low Priority A 6 Parameter	MTVBRLPA6P
– 0xFED000F4–0xFED000F7	MAPLE TVPE BD Ring Low Priority B 6 Parameter	MTVBRLPB6P
– 0xFED000F8–0xFED000FB	MAPLE TVPE BD Ring Low Priority A 7 Parameter	MTVBRLPA7P
– 0xFED000FC–0xFED000FF	MAPLE TVPE BD Ring Low Priority B 7 Parameter	MTVBRLPB7P
– 0xFED00100–0xFED00103	MAPLE FFTPE BD Ring High Priority A 0 Parameter	MFFBRHPA0P
– 0xFED00104–0xFED00107	MAPLE FFTPE BD Ring High Priority B 0 Parameter	MFFBRHPB0P
– 0xFED00108–0xFED0010B	MAPLE FFTPE BD Ring High Priority A 1 Parameter	MFFBRHPA1P
– 0xFED0010C–0xFED0010F	MAPLE FFTPE BD Ring High Priority B 1 Parameter	MFFBRHPB1P
– 0xFED00110–0xFED00113	MAPLE FFTPE BD Ring High Priority A 2 Parameter	MFFBRHPA2P
– 0xFED00114–0xFED00117	MAPLE FFTPE BD Ring High Priority B 2 Parameter	MFFBRHPB2P
– 0xFED00118–0xFED0011B	MAPLE FFTPE BD Ring High Priority A 3 Parameter	MFFBRHPA3P
– 0xFED0011C–0xFED0011F	MAPLE FFTPE BD Ring High Priority B 3 Parameter	MFFBRHPB3P
– 0xFED00120–0xFED00123	MAPLE FFTPE BD Ring High Priority A 4 Parameter	MFFBRHPA4P
– 0xFED00124–0xFED00127	MAPLE FFTPE BD Ring High Priority B 4 Parameter	MFFBRHPB4P
– 0xFED00128–0xFED0012B	MAPLE FFTPE BD Ring High Priority A 5 Parameter	MFFBRHPA5P
– 0xFED0012C–0xFED0012F	MAPLE FFTPE BD Ring High Priority B 5 Parameter	MFFBRHPB5P
– 0xFED00130–0xFED00133	MAPLE FFTPE BD Ring High Priority A 6 Parameter	MFFBRHPA6P
– 0xFED00134–0xFED00137	MAPLE FFTPE BD Ring High Priority B 6 Parameter	MFFBRHPB6P
– 0xFED00138–0xFED0013B	MAPLE FFTPE BD Ring High Priority A 7 Parameter	MFFBRHPA7P
– 0xFED0013C–0xFED0013F	MAPLE FFTPE BD Ring High Priority B 7 Parameter	MFFBRHPB7P
– 0xFED00140–0xFED00143	MAPLE FFTPE BD Ring Low Priority A 0 Parameter	MFFBRLPA0P
– 0xFED00144–0xFED00147	MAPLE FFTPE BD Ring Low Priority B 0 Parameter	MFFBRLPB0P
– 0xFED00148–0xFED0014B	MAPLE FFTPE BD Ring Low Priority A 1 Parameter	MFFBRLPA1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED0014C–0xFED0014F	MAPLE FFTPE BD Ring Low Priority B 1 Parameter	MFFBRLPB1P
– 0xFED00150–0xFED00153	MAPLE FFTPE BD Ring Low Priority A 2 Parameter	MFFBRLPA2P
– 0xFED00154–0xFED00157	MAPLE FFTPE BD Ring Low Priority B 2 Parameter	MFFBRLPB2P
– 0xFED00158–0xFED0015B	MAPLE FFTPE BD Ring Low Priority A 3 Parameter	MFFBRLPA3P
– 0xFED0015C–0xFED0015F	MAPLE FFTPE BD Ring Low Priority B 3 Parameter	MFFBRLPB3P
– 0xFED00160–0xFED00163	MAPLE FFTPE BD Ring Low Priority A 4 Parameter	MFFBRLPA4P
– 0xFED00164–0xFED00167	MAPLE FFTPE BD Ring Low Priority B 4 Parameter	MFFBRLPB4P
– 0xFED00168–0xFED0016B	MAPLE FFTPE BD Ring Low Priority A 5 Parameter	MFFBRLPA5P
– 0xFED0016C–0xFED0016F	MAPLE FFTPE BD Ring Low Priority B 5 Parameter	MFFBRLPB5P
– 0xFED00170–0xFED00173	MAPLE FFTPE BD Ring Low Priority A 6 Parameter	MFFBRLPA6P
– 0xFED00174–0xFED00177	MAPLE FFTPE BD Ring Low Priority B 6 Parameter	MFFBRLPB6P
– 0xFED00178–0xFED0017B	MAPLE FFTPE BD Ring Low Priority A 7 Parameter	MFFBRLPA7P
– 0xFED0017C–0xFED0017F	MAPLE FFTPE BD Ring Low Priority B 7 Parameter	MFFBRLPB7P
– 0xFED00180–0xFED00183	MAPLE DFTPE BD Ring High Priority A 0 Parameter	MDFBRHPA0P
– 0xFED00184–0xFED00187	MAPLE DFTPE BD Ring High Priority B 0 Parameter	MDFBRHPB0P
– 0xFED00188–0xFED0018B	MAPLE DFTPE BD Ring High Priority A 1 Parameter	MDFBRHPA1P
– 0xFED0018C–0xFED0018F	MAPLE DFTPE BD Ring High Priority B 1 Parameter	MDFBRHPB1P
– 0xFED00190–0xFED00183	MAPLE DFTPE BD Ring High Priority A 2 Parameter	MDFBRHPA2P
– 0xFED00194–0xFED00197	MAPLE DFTPE BD Ring High Priority B 2 Parameter	MDFBRHPB2P
– 0xFED00198–0xFED0019B	MAPLE DFTPE BD Ring High Priority A 3 Parameter	MDFBRHPA3P
– 0xFED0019C–0xFED0019F	MAPLE DFTPE BD Ring High Priority B 3 Parameter	MDFBRHPB3P
– 0xFED001A0–0xFED001A3	MAPLE DFTPE BD Ring High Priority A 4 Parameter	MDFBRHPA4P
– 0xFED001A4–0xFED001A7	MAPLE DFTPE BD Ring High Priority B 4 Parameter	MDFBRHPB4P
– 0xFED001A8–0xFED001AB	MAPLE DFTPE BD Ring High Priority A 5 Parameter	MDFBRHPA5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED001AC–0xFED001AF	MAPLE DFTPE BD Ring High Priority B 5 Parameter	MDFBRHPB5P
– 0xFED001B0–0xFED001B3	MAPLE DFTPE BD Ring High Priority A 6 Parameter	MDFBRHPA6P
– 0xFED001B4–0xFED001B7	MAPLE DFTPE BD Ring High Priority B 6 Parameter	MDFBRHPB6P
– 0xFED001B8–0xFED001BB	MAPLE DFTPE BD Ring High Priority A 7 Parameter	MDFBRHPA7P
– 0xFED001BC–0xFED001BF	MAPLE DFTPE BD Ring High Priority B 7 Parameter	MDFBRHPB7P
– 0xFED001C0–0xFED001C3	MAPLE DFTPE BD Ring Low Priority A 0 Parameter	MDFBRLPA0P
– 0xFED001C4–0xFED001C7	MAPLE DFTPE BD Ring Low Priority B 0 Parameter	MDFBRLPB0P
– 0xFED001C8–0xFED001CB	MAPLE DFTPE BD Ring Low Priority A 1 Parameter	MDFBRLPA1P
– 0xFED001CC–0xFED001CF	MAPLE DFTPE BD Ring Low Priority B 1 Parameter	MDFBRLPB1P
– 0xFED001D0–0xFED001D3	MAPLE DFTPE BD Ring Low Priority A 2 Parameter	MDFBRLPA2P
– 0xFED001D4–0xFED001D7	MAPLE DFTPE BD Ring Low Priority B 2 Parameter	MDFBRLPB2P
– 0xFED001D8–0xFED001DB	MAPLE DFTPE BD Ring Low Priority A 3 Parameter	MDFBRLPA3P
– 0xFED001DC–0xFED001DF	MAPLE DFTPE BD Ring Low Priority B 3 Parameter	MDFBRLPB3P
– 0xFED001E0–0xFED001E3	MAPLE DFTPE BD Ring Low Priority A 4 Parameter	MDFBRLPA4P
– 0xFED001E4–0xFED001E7	MAPLE DFTPE BD Ring Low Priority B 4 Parameter	MDFBRLPB4P
– 0xFED001E8–0xFED001EB	MAPLE DFTPE BD Ring Low Priority A 5 Parameter	MDFBRLPA5P
– 0xFED001EC–0xFED001EF	MAPLE DFTPE BD Ring Low Priority B 5 Parameter	MDFBRLPB5P
– 0xFED001F0–0xFED001F3	MAPLE DFTPE BD Ring Low Priority A 6 Parameter	MDFBRLPA6P
– 0xFED001F4–0xFED001F7	MAPLE DFTPE BD Ring Low Priority B 6 Parameter	MDFBRLPB6P
– 0xFED001F8–0xFED001FB	MAPLE DFTPE BD Ring Low Priority A 7 Parameter	MDFBRLPA7P
– 0xFED001FC–0xFED001FF	MAPLE DFTPE BD Ring Low Priority B 7 Parameter	MDFBRLPB7P
– 0xFED00200–0xFED00203	MAPLE CRCPE BD Ring High Priority A 0 Parameter	MCRCBRHPA0P
– 0xFED00204–0xFED00207	MAPLE CRCPE BD Ring High Priority B 0 Parameter	MCRCBRHPB0P
– 0xFED00208–0xFED0020B	MAPLE CRCPE BD Ring High Priority A 1 Parameter	MCRCBRHPA1P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED0020C–0xFED0020F	MAPLE CRCPE BD Ring High Priority B 1 Parameter	MCRCBRHPB1P
– 0xFED00210–0xFED00213	MAPLE CRCPE BD Ring High Priority A 2 Parameter	MCRCBRHPA2P
– 0xFED00214–0xFED00217	MAPLE CRCPE BD Ring High Priority B 2 Parameter	MCRCBRHPB2P
– 0xFED00218–0xFED0021B	MAPLE CRCPE BD Ring High Priority A 3 Parameter	MCRCBRHPA3P
– 0xFED0021C–0xFED0021F	MAPLE CRCPE BD Ring High Priority B 3 Parameter	MCRCBRHPB3P
– 0xFED00220–0xFED00223	MAPLE CRCPE BD Ring High Priority A 4 Parameter	MCRCBRHPA4P
– 0xFED00224–0xFED00227	MAPLE CRCPE BD Ring High Priority B 4 Parameter	MCRCBRHPB4P
– 0xFED00228–0xFED0022B	MAPLE CRCPE BD Ring High Priority A 5 Parameter	MCRCBRHPA5P
– 0xFED0022C–0xFED0022F	MAPLE CRCPE BD Ring High Priority B 5 Parameter	MCRCBRHPB5P
– 0xFED00230–0xFED00233	MAPLE CRCPE BD Ring High Priority A 6 Parameter	MCRCBRHPA6P
– 0xFED00234–0xFED00237	MAPLE CRCPE BD Ring High Priority B 6 Parameter	MCRCBRHPB6P
– 0xFED00238–0xFED0023B	MAPLE CRCPE BD Ring High Priority A 7 Parameter	MCRCBRHPA7P
– 0xFED0023C–0xFED0023F	MAPLE CRCPE BD Ring High Priority B 7 Parameter	MCRCBRHPB7P
– 0xFED00240–0xFED00243	MAPLE CRCPE BD Ring Low Priority A 0 Parameter	MCRCBRLPA0P
– 0xFED00244–0xFED00247	MAPLE CRCPE BD Ring Low Priority B 0 Parameter	MCRCBRLPB0P
– 0xFED00248–0xFED0024B	MAPLE CRCPE BD Ring Low Priority A 1 Parameter	MCRCBRLPA1P
– 0xFED0024C–0xFED0024F	MAPLE CRCPE BD Ring Low Priority B 1 Parameter	MCRCBRLPB1P
– 0xFED00250–0xFED00253	MAPLE CRCPE BD Ring Low Priority A 2 Parameter	MCRCBRLPA2P
– 0xFED00254–0xFED00257	MAPLE CRCPE BD Ring Low Priority B 2 Parameter	MCRCBRLPB2P
– 0xFED00258–0xFED0025B	MAPLE CRCPE BD Ring Low Priority A 3 Parameter	MCRCBRLPA3P
– 0xFED0025C–0xFED0025F	MAPLE CRCPE BD Ring Low Priority B 3 Parameter	MCRCBRLPB3P
– 0xFED00260–0xFED00263	MAPLE CRCPE BD Ring Low Priority A 4 Parameter	MCRCBRLPA4P
– 0xFED00264–0xFED00267	MAPLE CRCPE BD Ring Low Priority B 4 Parameter	MCRCBRLPB4P
– 0xFED00268–0xFED0026B	MAPLE CRCPE BD Ring Low Priority A 5 Parameter	MCRCBRLPA5P

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED0026C–0xFED0026F	MAPLE CRCPE BD Ring Low Priority B 5 Parameter	MCRCBRLPB5P
– 0xFED00270–0xFED00273	MAPLE CRCPE BD Ring Low Priority A 6 Parameter	MCRCBRLPA6P
– 0xFED00274–0xFED00277	MAPLE CRCPE BD Ring Low Priority B 6 Parameter	MCRCBRLPB6P
– 0xFED00278–0xFED0027B	MAPLE CRCPE BD Ring Low Priority A 7 Parameter	MCRCBRLPA7P
– 0xFED0027C–0xFED0027F	MAPLE CRCPE BD Ring Low Priority B 7 Parameter	MCRCBRLPB7P
– 0xFED00280–0xFED002FF	reserved	
– 0xFED00300–0xFED00303	MAPLE Turbo Stop Criteria Configuration Parameter	MTSCCP
– 0xFED00304–0xFED0037F	reserved	
– 0xFED00380–0xFED00383	MAPLE Turbo Viterbi Puncturing Vector 0 High Configuration Parameter	MTVPV0HCP
– 0xFED00384–0xFED00387	MAPLE Turbo Viterbi Puncturing Vector 0 Low Configuration Parameter	MTVPV0LCP
– 0xFED00388–0xFED0038B	MAPLE Turbo Viterbi Puncturing Vector 1 High Configuration Parameter	MTVPV1HCP
– 0xFED0038C–0xFED0038F	MAPLE Turbo Viterbi Puncturing Vector 1 Low Configuration Parameter	MTVPV1LCP
– 0xFED00390–0xFED00393	MAPLE Turbo Viterbi Puncturing Vector 2 High Configuration Parameter	MTVPV2HCP
– 0xFED00394–0xFED00397	MAPLE Turbo Viterbi Puncturing Vector 2 Low Configuration Parameter	MTVPV2LCP
– 0xFED00398–0xFED0039B	MAPLE Turbo Viterbi Puncturing Vector 3 High Configuration Parameter	MTVPV3HCP
– 0xFED0039C–0xFED0039F	MAPLE Turbo Viterbi Puncturing Vector 3 Low Configuration Parameter	MTVPV3LCP
– 0xFED003A0–0xFED003A3	MAPLE Turbo Viterbi Puncturing Vector 4 High Configuration Parameter	MTVPV4HCP
– 0xFED003A4–0xFED003A7	MAPLE Turbo Viterbi Puncturing Vector 4 Low Configuration Parameter	MTVPV4LCP
– 0xFED003A8–0xFED003AB	MAPLE Turbo Viterbi Puncturing Vector 5 High Configuration Parameter	MTVPV5HCP
– 0xFED003AC–0xFED003AF	MAPLE Turbo Viterbi Puncturing Vector 5 Low Configuration Parameter	MTVPV5LCP
– 0xFED003B0–0xFED003B3	MAPLE Turbo Viterbi Puncturing Vector 6 High Configuration Parameter	MTVPV6HCP
– 0xFED003B4–0xFED003B7	MAPLE Turbo Viterbi Puncturing Vector 6 Low Configuration Parameter	MTVPV6LCP
– 0xFED003B8–0xFED003BB	MAPLE Turbo Viterbi Puncturing Vector 7 High Configuration Parameter	MTVPV7HCP
– 0xFED003BC–0xFED003BF	MAPLE Turbo Viterbi Puncturing Vector 7 Low Configuration Parameter	MTVPV7LCP

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED003C0–0xFED003C3	MAPLE Turbo Viterbi Puncturing Vector 8 High Configuration Parameter	MTVPV8HCP
– 0xFED003C4–0xFED003C7	MAPLE Turbo Viterbi Puncturing Vector 8 Low Configuration Parameter	MTVPV8LCP
– 0xFED003C8–0xFED003CB	MAPLE Turbo Viterbi Puncturing Vector 9 High Configuration Parameter	MTVPV9HCP
– 0xFED003CC–0xFED003CF	MAPLE Turbo Viterbi Puncturing Vector 9 Low Configuration Parameter	MTVPV9LCP
– 0xFED003D0–0xFED003D3	MAPLE Turbo Viterbi Puncturing Period Configuration 0 Parameter	MTVPPC0P
– 0xFED003D4–0xFED003D7	MAPLE Turbo Viterbi Puncturing Period Configuration 1 Parameter	MTVPPC1P
– 0xFED003D8–0xFED003DB	MAPLE Turbo Viterbi Puncturing Period Configuration 2 Parameter	MTVPPC2P
– 0xFED003DC–0xFED003DF	reserved	
– 0xFED003E0–0xFED003E3	MAPLE Turbo Viterbi Polynomial Vector Set 0 Configuration 0 Parameter	MTVPVS0C0P
– 0xFED003E4–0xFED003E7	MAPLE Turbo Viterbi Polynomial Vector Set 0 Configuration 1 Parameter	MTVPVS0C1P
– 0xFED003E8–0xFED003EB	MAPLE Turbo Viterbi Polynomial Vector Set 1 Configuration 0 Parameter	MTVPVS1C0P
– 0xFED003EC–0xFED003EF	MAPLE Turbo Viterbi Polynomial Vector Set 1 Configuration 1 Parameter	MTVPVS1C1P
– 0xFED003F0–0xFED003F3	MAPLE Turbo Viterbi Polynomial Vector Set 2 Configuration 0 Parameter	MTVPVS2C0P
– 0xFED003F4–0xFED003F7	MAPLE Turbo Viterbi Polynomial Vector Set 2 Configuration 1 Parameter	MTVPVS2C1P
– 0xFED003F8–0xFED0043F	reserved	
– 0xFED00440–0xFED00443	MAPLE-B Turbo Total Performance Parameter	MTTPP
– 0xFED00444–0xFED00447	MAPLE-B Viterbi Total Performance Parameter	MVTPP
– 0xFED00448–0xFED0044B	MAPLE-B Total BLER Parameter	MTBP
– 0xFED0044C–0xFED0044F	MAPLE-B TVPE Turbo BDs Counter Parameter	MTBDCP
– 0xFED00450–0xFED00453	MAPLE-B FFT Total Performance Parameter	MFTPP
– 0xFED00454–0xFED00457	MAPLE-B FFTPE BDs Counter Parameter	MFBDCP
– 0xFED00458–0xFED0045B	MAPLE-B DFT Total Performance Parameter	MDTPP
– 0xFED0045C–0xFED0045F	MAPLE-B DFTPE BDs Counter Parameter	MDBDCP
– 0xFED00460–0xFED00463	MAPLE-B Profiling Enable Parameter	MPEP

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED00464–0xFED0047F	reserved	
– 0xFED00480–0xFED00483	Serial RapidIO Outbound RapidIO Doorbell Port0 Base Address Parameter	SORDP0BAP
– 0xFED00484–0xFED00487	Serial RapidIO Outbound RapidIO Doorbell Port1 Base Address Parameter	SORDP1BAP
– 0xFED00488–0xFED00480B	Hardware Semaphore Port0 Base Address Parameter	HSP0BAP
– 0xFED0048C–0xFED0048F	Hardware Semaphore Port1 Base Address Parameter	HSP1BAP
– 0xFED00490–0xFED00493	MAPLE-B Doorbell Hardware Semaphore ID Configuration Parameter	MDHSIDCP
– 0xFED00494–0xFED00497	MAPLE-B Doorbell General Configuration Parameter	MDGCP
– 0xFED00498–0xFED004FF	reserved	
– 0xFED00500–0xFED03FFF	reserved	
– 0xFED04000–0xFED06FFF	Buffer Descriptor (BD) Rings	
– 0xFED07000–0xFED1FFFF	reserved	
– 0xFED20000–0xFED3FFFF	reserved	
– 0xFED40000–0xFED5FFFF	TVPE (registers begin on **page 26-174**)	
– 0xFED40000–0xFED5DFFF	reserved	
– 0xFED5E000	TVPE Configuration 0 Register	TVPEC0R
– 0xFED5E004	reserved	
– 0xFED5E008	TVPE Symbol Identification 0 Configuration Register	TVSI0CR
– 0xFED5E00C	TVPE Symbol Identification 1 Configuration Register	TVSI1CR
– 0xFED5E010–0xFED5E023	TVPE Tail Symbol Identification X Configuration Register	TVTTSIxCR
– 0xFED5E024–0xFED5E03B	reserved	
– 0xFED5E03C	TVPE Aposteriori Quality Configuration Register	TVAQCR
– 0xFED5E040	Viterbi Polynomial Vector Generation 0 Configuration Register	TVVPVG0CR
– 0xFED5E044	TVPE Viterbi Polynomial Vector Generation 1 Configuration Register	TVVPVG1CR
– 0xFED5E048–0xFED5ECFF	reserved	
– 0xFED5ED00	TVPE Decoder Status Register	TVPESR
– 0xFED5ED10–0xFED5FFFF	reserved	
– 0xFED60000–0xFED7FFFF	FFTPE (registers begin on **page 26-185**)	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFED60000–0xFED67FFF	reserved	
– 0xFED68000–0xFED69FFF	Pre-Multiplication Memory	
– 0xFED6A000–0xFED7E13F	reserved	
– 0xFED7E140–0xFED7E148	FFTPE Data Size Registers	FFTPEDSRx
– 0xFED7E149–0xFED7E237	reserved	
– 0xFED7E238	FFTPE Status Register	FFTPESTR
– 0xFED7E23C	FFTPE Scaling Status Register	FFTPESCLSTR
– 0xFED7E240	FFTPE Saturation Status Register 0	FFTPESSTR0
– 0xFED7E244	FFTPE Saturation Status Register 1	FFTPESSTR1
– 0xFED7E248	FFTPE Saturation Status Register 2	FFTPESSTR2
– 0xFED7E24C–0xFED7FFFF	reserved	
– 0xFED80000–0xFED9FFFF	DFTPE (registers begin on **page 26-194**)	
– 0xFED80000–0xFED87FFF	reserved	
– 0xFED88000–0xFED89FFF	Pre-Multiplication Memory	
– 0xFED8A000–0xFED9E13F	reserved	
– 0xFED9E140–0xFED9E147	DFTPE Data Size Registers	DFTPEDSRx
– 0xFED9E148–0xFED9E237	reserved	
– 0xFED9E238	DFTPE Status Register	DFTPESTR
– 0xFED9E23C	DFTPE Scaling Status Register	DFTPESCLSTR
– 0xFED9E240	DFTPE Saturation Status Register 0	DFTPESSTR0
– 0xFED9E244	DFTPE Saturation Status Register 1	DFTPESSTR1
– 0xFED9E248	DFTPE Saturation Status Register 2	DFTPESSTR2
– 0xFED9E24C	DFTPE Saturation Status Register 3	DFTPESSTR3
– 0xFED9E250–0xFED9FFFF	reserved	
– 0xFEDA0000–0xFEDFFFFF	reserved	
• 0xFEE00000–0xFEE3FFFF	QUICC Engine Subsystem (Refer to the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM) for register, configuration, and programming details).	
– 0xFEE00000	IRAM Address Register	IADD
– 0xFEE00004	IRAM Data Register	IDATA
– 0xFEE00008–0xFEE0007F	reserved	
– 0xFEE00080	QUICC Engine Interrupt Configuration Register	CICR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFEE00084	QUICC Engine System Interrupt Vector Register	CIVEC
– 0xFEE00088	QUICC Engine Interrupt Pending Register	CRIPNR
– 0xFEE0008C	QUICC Engine System Interrupt Pending Register	CIPNR
– 0xFEE00090	QUICC Engine Interrupt Priority Register (XCC)	CIPXCC
– 0xFEE00094–0xFEE00097	reserved	
– 0xFEE00098	QUICC Engine Interrupt Priority Register (WCC)	CIPWCC
– 0xFEE0009C	QUICC Engine Interrupt Priority Register (ZCC)	CIPZCC
– 0xFEE000A0	QUICC Engine System Interrupt Mask Register	CIMR
– 0xFEE000A4	QUICC Engine RISC Interrupt Mask Register	CRIMR
– 0xFEE000A8	QUICC Engine System Interrupt Control Register	CICNR
– 0xFEE000AC–0xFEE000AF	reserved	
– 0xFEE000B0	QUICC Engine System Interrupt Priority Register for RISC Tasks A	CIPRTA
– 0xFEE000B4–0xFEE000BB	reserved	
– 0xFEE000BC	QUICC Engine System RISC Interrupt Control Register	CRICR
– 0xFEE000C0–0xFEE000DF	reserved	
– 0xFEE000E0	QUICC Engine High System Interrupt Vector Register	CHIVEC
– 0xFEE000E4–0xFEE000FF	reserved	
– 0xFEE00100	QUICC Engine Command Register	CECR
– 0xFEE00104–0xFEE00107	reserved	
– 0xFEE00108	QUICC Engine Command Data Register	CECDR
– 0xFEE0010C–0xFEE0011B	reserved	
– 0xFEE0011C	QUICC Engine Time-Stamp Control Register	CETSCR
– 0xFEE00120–0xFEE0012F	reserved	
– 0xFEE00130	QUICC Engine Virtual Task Event Register	CEVTER
– 0xFEE00134	QUICC Engine Virtual Task Mask Register	CEVTMR
– 0xFEE00138	QUICC Engine RAM Control Register	CERCR
– 0xFEE0013C–0xFEE001B7	reserved	
– 0xFEE001B8	QUICC Engine Microcode Revision Number Register	CEURNR
– 0xFEE001BC–0xFEE003FF	reserved	
– 0xFEE00400	CMX General Clock Route Register	CMXGCR
– 0xFEE00404–0xFEE0040F	reserved	
– 0xFEE00410	CMX Clock Route Register 1	CMXUCR1
– 0xFEE00414–0xFEE004DF	Reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFEE004E0	SPI Mode Register	SPIMODE
– 0xFEE004E4	SPI Event Register	SPIE
– 0xFEE004E8	SPI Mask Register	SPIM
– 0xFEE004EC	SPI Command Register	SPCOM
– 0xFEE004F0–0xFEE0064F	Reserved	
– 0xFEE00650	Baud-Rate Generator Configuration Register 5	BRGCR5
– 0xFEE00654	Baud-Rate Generator Configuration Register 6	BRGCR6
– 0xFEE00658	Baud-Rate Generator Configuration Register 7	BRGCR7
– 0xFEE0065C	Baud-Rate Generator Configuration Register 8	BRGCR8
– 0xFEE00660–0xFEE01FFF	reserved	
– 0xFEE02000	UCC 1 Mode Register	GUMR1
– 0xFEE02004	UCC 1 Protocol Specific Mode Register	UPSMR1
– 0xFEE02008	UCC 1 Transmit-on-Demand Register	UTODR1
– 0xFEE0200A–0xFEE0200F	reserved	
– 0xFEE02010	UCC 1 Event Register	UCCE1
– 0xFEE02014	UCC 1 Mask Register	UCCM1
– 0xFEE02018	UCC 1 Ethernet Transmitter Status Register	UCCS1
– 0xFEE02019–0xFEE0201F	reserved	
– 0xFEE02020	UCC 1 Receive FIFO Base	URFB1
– 0xFEE02024	UCC 1 Receive FIFO Size	URFS1
– 0xFEE02026–0xFEE02027	reserved	
– 0xFEE02028	UCC 1 Receive FIFO Emergency Threshold	URFET1
– 0xFEE0202A	UCC 1 Receive FIFO Special Emergency Threshold	URFSET1
– 0xFEE0202C	UCC 1 Transmit FIFO Base	UTFB1
– 0xFEE02030	UCC 1 Transmit FIFO Size	UTFS1
– 0xFEE02032–0xFEE02033	reserved	
– 0xFEE02034	UCC 1 Transmit FIFO Emergency Threshold	UTFET1
– 0xFEE02036–0xFEE02037	reserved	
– 0xFEE02038	UCC 1 Transmit FIFO Transmit Threshold	UTFTT1
– 0xFEE0203A–0xFEE0203B	reserved	
– 0xFEE0203C	UCC 1 Transmit Polling Timer	UFPT1
– 0xFEE0203E–0xFEE0203F	reserved	
– 0xFEE02040	UCC 1 Retry Counter Register	URTRY1
– 0xFEE02044–0xFEE0208F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFEE02090	UCC 1 General Extended Mode Register	GUEMR1
– 0xFEE02094–0xFEE020FF	reserved	
– 0xFEE02100	Ethernet 1 MAC Configuration Register 1	E1MACCFG1
– 0xFEE02104	Ethernet 1 MAC Configuration Register 2	E1MACCFG2
– 0xFEE02108	Ethernet 1 Interframe Gap Register	E1IPGFG
– 0xFEE0210A–0xFEE0210B	reserved	
– 0xFEE0210C	Ethernet 1 Half-Duplex Register	HAFDUP1
– 0xFEE02110–0xFEE0211F	reserved	
– 0xFEE02120	Ethernet 1 MII Management Configuration Register	MIIMCFG1
– 0xFEE02124	Ethernet 1 MII Management Command Register	MIIMCOM1
– 0xFEE02128	Ethernet 1 MII Management Address Register	MIIMADD1
– 0xFEE0212C	Ethernet 1 MII Management Control Register	MIIMCON1
– 0xFEE02130	Ethernet 1 MII Management Status Register	MIIMSTAT1
– 0xFEE02134	Ethernet 1 MII Management Indication Register	MIIMIND1
– 0xFEE0213C	Ethernet 1 Interface Status Register	IFSTAT1
– 0xFEE02140	Ethernet 1 Station Address Pt. 1 Register	E1MACSTNADDDR1
– 0xFEE02144	Ethernet 1 Station Address Pt. 2 Register	E1MACSTNADDR2
– 0xFEE02148–0xFEE0214F	reserved	
– 0xFEE02150	Ethernet 1 MAC Parameter Register	UEMPR1
– 0xFEE02154	Ethernet 1 Ten-Bit Interface Physical Address Register	UTBIPAR1
– 0xFEE02158	Ethernet 1 Statistical Control Register	UESCR1
– 0xFEE0215C–0xFEE0217F	reserved	
– 0xFEE02180	Ethernet 1 Tx 64-byte Frames	E1TX64
– 0xFEE02184	Ethernet 1 Tx 65- to 127-byte Frames	E1TX127
– 0xFEE02188	Ethernet 1 Tx 128- to 255-byte Frames	E1TX255
– 0xFEE0218C	Ethernet 1 Rx 64-byte Frames	E1RX64
– 0xFEE02190	Ethernet 1 Rx 65- to 127-byte Frames	E1RX127
– 0xFEE02194	Ethernet 1 Rx 128- to 255-byte Frames	E1RX255
– 0xFEE02198	Ethernet 1 Octet Transmitted OK	E1TXOK
– 0xFEE0219C	Ethernet 1 Tx Pause Frames	E1TXCF
– 0xFEE021A0	Ethernet 1 Multicast Frame Transmitted OK	E1TMCA
– 0xFEE021A4	Ethernet 1 Broadcast Frames Transmitted OK	E1TBCA
– 0xFEE021A8	Ethernet 1 Number of Frames Received OK	E1RXFOK
– 0xFEE021AC	Ethernet 1 Rx Octets OK	E1RBYT
– 0xFEE021B0	Ethernet 1 Rx Octets	E1RXBOK
– 0xFEE021B4	Ethernet 1 Multicast Frame Received OK	E1RMCA
– 0xFEE021B8	Ethernet 1 Broadcast Frames Received OK	E1RBCA
– 0xFEE021BC	Ethernet 1 Statistic Counters Carry Register	E1SCAR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFEE021C0	Ethernet 1 Statistic Counters Carry Mask Register	E1SCAM
– 0xFEE021C4–0xFEE021FF	reserved	
– 0xFEE02200	UCC 3 General Mode Register	GUMR3
– 0xFEE02204	UCC 3 Protocol Specific Mode Register	UPSMR3
– 0xFEE02208	UCC 3 Transmit-on-Demand Register	UTODR3
– 0xFEE0220A–0xFEE0220F	reserved	
– 0xFEE02210	UCC 3 Event Register	UCCE3
– 0xFEE02214	UCC 3 Mask Registers	UCCM3
– 0xFEE02218	UCC 3 Status Register	UCCS3
– 0xFEE02219–0xFEE0221F	reserved	
– 0xFEE02220	UCC 3 Receive FIFO Base	URFB3
– 0xFEE02224	UCC 3 Receive FIFO Size	URFS3
– 0xFEE02226–0xFEE02227	reserved	
– 0xFEE02228	UCC 3 Receive FIFO Emergency Threshold	URFET3
– 0xFEE0222A	UCC 3 Receive FIFO Special Emergency Threshold	URFSET3
– 0xFEE0222C	UCC 3 Transmit FIFO Base	UTFB3
– 0xFEE02230	UCC 3 Transmit FIFO Size	UTFS3
– 0xFEE02232–0xFEE02233	reserved	
– 0xFEE02234	UCC 3 Transmit FIFO Emergency Threshold	UTFET3
– 0xFEE02236–0xFEE02237	reserved	
– 0xFEE02238	UCC 3 Transmit FIFO Transmit Threshold	UTFTT3
– 0xFEE0223A–0xFEE0223B	reserved	
– 0xFEE0223C	UCC 3 Transmit Polling Timer	UFPT3
– 0xFEE0223E–0xFEE0223F	reserved	
– 0xFEE02240	UCC 3 Retry Counter	URTRY3
– 0xFEE02244–0xFEE0228F	reserved	
– 0xFEE02290	UCC 3 General Extended Mode Register	GUEMR3
– 0xFEE02294–0xFEE022FF	reserved	
– 0xFEE02300	Ethernet 2 MAC Configuration Register 1	E2MACCFG1
– 0xFEE02304	Ethernet 2 MAC Configuration Register 2	E2MACCFG2
– 0xFEE02308	Ethernet 2 Interframe Gap Register	E2IPGIFG
– 0xFEE0230A–0xFEE0230B	reserved	
– 0xFEE0230C	Ethernet 2 Half-Duplex Register	HAFDUP3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFEE02310–0xFEE0231F	reserved	
– 0xFEE02320	Ethernet 2 MII Management Configuration Register	MIIMCFG3
– 0xFEE02324	Ethernet 2 MII Management Command Register	MIIMCOM3
– 0xFEE02328	Ethernet 2 MII Management Address Register	MIIMADD3
– 0xFEE0232C	Ethernet 2 MII Management Control Register	MIIMCON3
– 0xFEE02330	Ethernet 2 MII Management Status Register	MIIMSTAT3
– 0xFEE02334	Ethernet 2 MII Management Indication Register	MIIMIND3
– 0xFEE02338–0xFEE0233B	reserved	
– 0xFEE0233C	Ethernet 2 Interface Status Register	IFSTAT3
– 0xFEE02340	Ethernet 2 Station Address Pt. 1 Register	E2MACSTNADDR1
– 0xFEE02344	Ethernet 2 Station Address Pt. 2 Register	E2MACSTNADDR2
– 0xFEE02348–0xFEE0234F	reserved	
– 0xFEE02350	Ethernet 2 Ethernet MAC Parameter Register	UEMPR3
– 0xFEE02354	Ethernet 2 Ten-Bit Interface Physical Address Register	TBIPAR3
– 0xFEE02358	Ethernet 2 Ethernet Statistical Control Register	UESCR3
– 0xFEE0235C–0xFEE0237F	reserved	
– 0xFEE02380	Ethernet 2 Tx 64-byte Frames	E2TX64
– 0xFEE02384	Ethernet 2 Tx 65- to 127-byte Frames	E2TX127
– 0xFEE02388	Ethernet 2 Tx 128- to 255-byte Frames	E2TX255
– 0xFEE0238C	Ethernet 2 Rx 64-byte Frames	E2RX64
– 0xFEE02390	Ethernet 2 Rx 65- to 127-byte Frames	E2RX127
– 0xFEE02394	Ethernet 2 Rx 128- to 255-byte Frames	E2RX255
– 0xFEE02398	Ethernet 2 Octet Transmitted OK	E2TXOK
– 0xFEE0239C	Ethernet 2 Tx Pause Frames	E2TXCF
– 0xFEE023A0	Ethernet 2 Multicast Frame Transmitted OK	E2TMCA
– 0xFEE023A4	Ethernet 2 Broadcast Frames Transmitted OK	E2TBCA
– 0xFEE023A8	Ethernet 2 Number of Frames Received OK	E2RXFOK
– 0xFEE023AC	Ethernet 2 Rx Octets OK	E2RBYT
– 0xFEE023B0	Ethernet 2 Rx Octets	E2RXBOK
– 0xFEE023B4	Ethernet 2 Multicast Frame Received OK	E2RMCA
– 0xFEE023B8	Ethernet 2 Broadcast Frames Received OK	E2RBCA
– 0xFEE023BC	Ethernet 2 Statistic Counters Carry Register	E2SCAR
– 0xFEE023C0	Ethernet 2 Statistic Counters Carry Mask Register	E2SCAM
– 0xFEE023C4–0xFEE03FFF	reserved	
– 0xFEE04000	Serial DMA Status Register	SDSR
– 0xFEE04004	Serial DMA Mode Register	SDMR
– 0xFEE04008	Serial DMA Threshold Register	SDTR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFEE0400C–0xFEE0400F	reserved	
– 0xFEE04010	Serial DMA Hysteresis Register	SDHY
– 0xFEE04014–0xFEE04017	reserved	
– 0xFEE04018	Serial DMA Address Register	SDTA
– 0xFEE0401C–0xFEE0401F	reserved	
– 0xFEE04020	Serial DMA MSNUM Register	SDTM
– 0xFEE04024–0xFEE04037	reserved	
– 0xFEE04038	Serial DMA Address Qualify Register	SDAQR
– 0xFEE0403C	Serial DMA Address Qualify Mask Register	SDAQMR
– 0xFEE04040–0xFEE04043	reserved	
– 0xFEE04044	Serial DMA Temporary Buffer Base in Multi-User RAM Value	SDEBCR
– 0xFEE04048–0xFEE07FFF	reserved	
– 0xFEE08000–0xFEE0FFFF	RAM space reserved	
– 0xFEE10000–0xFEE1BFFF	Multi-User RAM	
– 0xFEE1C000–0xFEE3FFFF	reserved	
• 0xFEE40000–0xFEEFFFFF	reserved (for QUICC Engine subsystem)	
• 0xFEF00000–0xFEF17FFF	Boot ROM	
• 0xFEF18000–0xFEFFFFFF	reserved	
0xFF000000–0xFFF0FFFF	DSP core subsystem internal memory space. See the *MSC8156E SC3850 DSP Core Subsystem Reference Manual* for details.	
0xFFF10000–0xFFFFEFFF	CCSR	
• 0xFFF10000–0xFFF103FF	DMA (see **Chapter 14**, *Direct Memory Access (DMA) Controller*)	
– 0xFFF10000	DMA Buffer Descriptor Base Register 0	DMABDBR0
– 0xFFF10004	DMA Buffer Descriptor Base Register 1	DMABDBR1
– 0xFFF10008	DMA Buffer Descriptor Base Register 2	DMABDBR2
– 0xFFF1000C	DMA Buffer Descriptor Base Register 3	DMABDBR3
– 0xFFF10010	DMA Buffer Descriptor Base Register 4	DMABDBR4
– 0xFFF10014	DMA Buffer Descriptor Base Register 5	DMABDBR5
– 0xFFF10018	DMA Buffer Descriptor Base Register 6	DMABDBR6
– 0xFFF1001C	DMA Buffer Descriptor Base Register 7	DMABDBR7
– 0xFFF10020	DMA Buffer Descriptor Base Register 8	DMABDBR8
– 0xFFF10024	DMA Buffer Descriptor Base Register 9	DMABDBR9

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF10028	DMA Buffer Descriptor Base Register 10	DMABDBR10
– 0xFFF1002C	DMA Buffer Descriptor Base Register 11	DMABDBR11
– 0xFFF10030	DMA Buffer Descriptor Base Register 12	DMABDBR12
– 0xFFF10034	DMA Buffer Descriptor Base Register 13	DMABDBR13
– 0xFFF10038	DMA Buffer Descriptor Base Register 14	DMABDBR14
– 0xFFF1003C	DMA Buffer Descriptor Base Register 15	DMABDBR15
– 0xFFF10040–0xFFF100FF	Reserved	
– 0xFFF10100	DMA Channel Configuration Register 0	DMACHCR0
– 0xFFF10104	DMA Channel Configuration Register 1	DMACHCR1
– 0xFFF10108	DMA Channel Configuration Register 2	DMACHCR2
– 0xFFF1010C	DMA Channel Configuration Register 3	DMACHCR3
– 0xFFF10110	DMA Channel Configuration Register 4	DMACHCR4
– 0xFFF10114	DMA Channel Configuration Register 5	DMACHCR5
– 0xFFF10118	DMA Channel Configuration Register 6	DMACHCR6
– 0xFFF1011C	DMA Channel Configuration Register 7	DMACHCR7
– 0xFFF10120	DMA Channel Configuration Register 8	DMACHCR8
– 0xFFF10124	DMA Channel Configuration Register 9	DMACHCR9
– 0xFFF10128	DMA Channel Configuration Register 10	DMACHCR10
– 0xFFF1012C	DMA Channel Configuration Register 11	DMACHCR11
– 0xFFF10130	DMA Channel Configuration Register 12	DMACHCR12
– 0xFFF10134	DMA Channel Configuration Register 13	DMACHCR13
– 0xFFF10138	DMA Channel Configuration Register 14	DMACHCR14
– 0xFFF1013C	DMA Channel Configuration Register 15	DMACHCR15
– 0xFFF10140–0xFFF101FF	Reserved	
– 0xFFF10200	DMA Global Configuration Register	DMAGCR
– 0xFFF10204	DMA Channel Enable Register	DMACHER
– 0xFFF10208–0xFFF1020B	Reserved	
– 0xFFF1020C	DMA Channel Disable Register	DMACHDR
– 0xFFF10210–0xFFF10213	Reserved	
– 0xFFF10214	DMA Channel Freeze Register	DMACHFR
– 0xFFF10218–0xFFF10223	Reserved	
– 0xFFF10224	DMA Channel Defrost Register	DMACHDFR
– 0xFFF10228–0xFFF10233	Reserved	
– 0xFFF10234	DMA Time-To-Deadline Register 0	DMAEDFTDL0
– 0xFFF10238	DMA Time-To-Deadline Register 1	DMAEDFTDL1
– 0xFFF1023C	DMA Time-To-Deadline Register 2	DMAEDFTDL2
– 0xFFF10240	DMA Time-To-Deadline Register 3	DMAEDFTDL3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF10244	DMA Time-To-Deadline Register 4	DMAEDFTDL4
– 0xFFF10248	DMA Time-To-Deadline Register 5	DMAEDFTDL5
– 0xFFF1024C	DMA Time-To-Deadline Register 6	DMAEDFTDL6
– 0xFFF10250	DMA Time-To-Deadline Register 7	DMAEDFTDL7
– 0xFFF10254	DMA Time-To-Deadline Register 8	DMAEDFTDL8
– 0xFFF10258	DMA Time-To-Deadline Register 9	DMAEDFTDL9
– 0xFFF1025C	DMA Time-To-Deadline Register 10	DMAEDFTDL10
– 0xFFF10260	DMA Time-To-Deadline Register 11	DMAEDFTDL11
– 0xFFF10264	DMA Time-To-Deadline Register 12	DMAEDFTDL12
– 0xFFF10268	DMA Time-To-Deadline Register 13	DMAEDFTDL13
– 0xFFF1026C	DMA Time-To-Deadline Register 14	DMAEDFTDL14
– 0xFFF10270	DMA Time-To-Deadline Register 15	DMAEDFTDL15
– 0xFFF10274–0xFFF10333	Reserved	
– 0xFFF10334	DMA EDF Control Register	DMAEDFCTRL
– 0xFFF10338	DMA EDF Mask Register	DMAEDFMR
– 0xFFF1033C–0xFFF1033f	Reserved	
– 0xFFF10340	DMA EDF Mask Update Register	DMAEDFMUR
– 0xFFF10344	DMA EDF Status Register	DMAEDFSTR
– 0xFFF10348–0xFFF1034B	Reserved	
– 0xFFF1034C	DMA Mask Register	DMAMR
– 0xFFF10350–0xFFF1035B	Reserved	
– 0xFFF1035C	DMA Mask Update Register	DMAMUR
– 0xFFF10360	DMA Destination Status Register	DMASTR
– 0xFFF10364–0xFFF1036F	Reserved	
– 0xFFF10370	DMA Error Register	DMAERR
– 0xFFF10374	DMA Debug Event Status Register	DMADESR
– 0xFFF10378	DMA Local Profiling Configuration Register	DMALPCR
– 0xFFF1037C	DMA Round Robin Priority Group Update Register	DMARRPGUR
– 0xFFF10380	DMA Channel Active Status Register	DMACHASTR
– 0xFFF10384–0xFFF10387	Reserved	
– 0xFFF10388	DMA Channel Freeze Status Register	DMACHFSTR
– 0xFFF1038C–0xFFF103FF	reserved	
• 0xFFF10400–0xFFF17FFF	reserved	
• 0xFFF18000–0xFFF18FFF	CLASS (see **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)*)	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF18000–0xFFF1801F	Reserved	
– 0xFFF18020	CLASS MBus Target Configuration Register 1	C0MTCR1
– 0xFFF18024–0xFFF1803F	Reserved	
– 0xFFF18040	CLASS MBus Target Configuration Register 2	C0MTCR2
– 0xFFF18044–0xFFF1805F	Reserved	
– 0xFFF18060	CLASS MBus Target Configuration Register 3	C0MTCR3
– 0xFFF18064–0xFFF1807F	Reserved	
– 0xFFF18080	CLASS MBus Target Configuration Register 4	C0MTCR4
– 0xFFF18084–0xFFF1808F	Reserved	
– 0xFFF18090	CLASS MBus Target Configuration Register 5	C0MTCR5
– 0xFFF18094–0xFFF1809F	Reserved	
– 0xFFF180A0	CLASS MBus Target Configuration Register 6	C0MTCR6
– 0xFFF180A4–0xFFF180AF	Reserved	
– 0xFFF180B0	CLASS MBus Target Configuration Register 7	C0MTCR7
– 0xFFF180B4–0xFFF187FF	Reserved	
– 0xFFF18800	CLASS Priority Mapping Register 0	C0PMR0
– 0xFFF18804	CLASS Priority Mapping Register 1	C0PMR1
– 0xFFF18808	CLASS Priority Mapping Register 2	C0PMR2
– 0xFFF1880C	CLASS Priority Mapping Register 3	C0PMR3
– 0xFFF18810	CLASS Priority Mapping Register 4	C0PMR4
– 0xFFF18814	CLASS Priority Mapping Register 5	C0PMR5
– 0xFFF18818	CLASS Priority Mapping Register 6	C0PMR6
– 0xFFF1881C	CLASS Priority Mapping Register 7	C0PMR7
– 0xFFF18820	CLASS Priority Mapping Register 8	C0PMR8
– 0xFFF18824	CLASS Priority Mapping Register 9	C0PMR9
– 0xFFF18828	CLASS Priority Mapping Register 10	C0PMR10
– 0xFFF1882C	CLASS Priority Mapping Register 11	C0PMR11
– 0xFFF18830–0xFFF1883F	reserved	
– 0xFFF18840	CLASS Priority Auto Upgrade Value Register 0	C0PAVR0
– 0xFFF18844	CLASS Priority Auto Upgrade Value Register 1	C0PAVR1
– 0xFFF18848	CLASS Priority Auto Upgrade Value Register 2	C0PAVR2
– 0xFFF1884C	CLASS Priority Auto Upgrade Value Register 3	C0PAVR3
– 0xFFF18850	CLASS Priority Auto Upgrade Value Register 4	C0PAVR4
– 0xFFF18854	CLASS Priority Auto Upgrade Value Register 5	C0PAVR5
– 0xFFF18858	CLASS Priority Auto Upgrade Value Register 6	C0PAVR6

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF1885C	CLASS Priority Auto Upgrade Value Register 7	C0PAVR7
– 0xFFF18860	CLASS Priority Auto Upgrade Value Register 8	C0PAVR8
– 0xFFF18864	CLASS Priority Auto Upgrade Value Register 9	C0PAVR9
– 0xFFF18868	CLASS Priority Auto Upgrade Value Register 10	C0PAVR10
– 0xFFF1886C	CLASS Priority Auto Upgrade Value Register 11	C0PAVR11
– 0xFFF18870–0xFFF1887F	reserved	
– 0xFFF18880	CLASS Priority Auto Upgrade Control Register 0	C0PACR0
– 0xFFF18884	CLASS Priority Auto Upgrade Control Register 1	C0PACR1
– 0xFFF18888	CLASS Priority Auto Upgrade Control Register 2	C0PACR2
– 0xFFF1888C	CLASS Priority Auto Upgrade Control Register 3	C0PACR3
– 0xFFF18890	CLASS Priority Auto Upgrade Control Register 4	C0PACR4
– 0xFFF18894	CLASS Priority Auto Upgrade Control Register 5	C0PACR5
– 0xFFF18898	CLASS Priority Auto Upgrade Control Register 6	C0PACR6
– 0xFFF1889C	CLASS Priority Auto Upgrade Control Register 7	C0PACR7
– 0xFFF188A0	CLASS Priority Auto Upgrade Control Register 8	C0PACR8
– 0xFFF188A4	CLASS Priority Auto Upgrade Control Register 9	C0PACR9
– 0xFFF188A8	CLASS Priority Auto Upgrade Control Register 10	C0PACR10
– 0xFFF188AC	CLASS Priority Auto Upgrade Control Register 11	C0PACR11
– 0xFFF188B0–0xFFF1897F	reserved	
– 0xFFF18980	CLASS Error Address Register 0	C0EAR0
– 0xFFF18984	CLASS Error Address Register 1	C0EAR1
– 0xFFF18988	CLASS Error Address Register 2	C0EAR2
– 0xFFF1898C	CLASS Error Address Register 3	C0EAR3
– 0xFFF18990	CLASS Error Address Register 4	C0EAR4
– 0xFFF18994	CLASS Error Address Register 5	C0EAR5
– 0xFFF18998	CLASS Error Address Register 6	C0EAR6
– 0xFFF1899C	CLASS Error Address Register 7	C0EAR7
– 0xFFF189A0	CLASS Error Address Register 8	C0EAR8
– 0xFFF189A4	CLASS Error Address Register 9	C0EAR9
– 0xFFF189A8	CLASS Error Address Register 10	C0EAR10
– 0xFFF189AC	CLASS Error Address Register 11	C0EAR11
– 0xFFF189B0–0xFFF189BF	reserved	
– 0xFFF189C0	CLASS Error Extended Address Register 0	C0EEAR0
– 0xFFF189C4	CLASS Error Extended Address Register 1	C0EEAR1
– 0xFFF189C8	CLASS Error Extended Address Register 2	C0EEAR2
– 0xFFF189CC	CLASS Error Extended Address Register 3	C0EEAR3
– 0xFFF189D0	CLASS Error Extended Address Register 4	C0EEAR4
– 0xFFF189D4	CLASS Error Extended Address Register 5	C0EEAR5
– 0xFFF189D8	CLASS Error Extended Address Register 6	C0EEAR6

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF189DC	CLASS Error Extended Address Register 7	C0EEAR7
– 0xFFF189E0	CLASS Error Extended Address Register 8	C0EEAR8
– 0xFFF189E4	CLASS Error Extended Address Register 9	C0EEAR9
– 0xFFF189E8	CLASS Error Extended Address Register 10	C0EEAR10
– 0xFFF189EC	CLASS Error Extended Address Register 11	C0EEAR11
– 0xFFF189F0–0xFFF189FF	reserved	
– 0xFFF18A00	CLASS Initiator Profiling Configuration Register 0	C0IPCR0
– 0xFFF18A04	CLASS Initiator Profiling Configuration Register 1	C0IPCR1
– 0xFFF18A08	CLASS Initiator Profiling Configuration Register 2	C0IPCR2
– 0xFFF18A0C	CLASS Initiator Profiling Configuration Register 3	C0IPCR3
– 0xFFF18A10	CLASS Initiator Profiling Configuration Register 4	C0IPCR4
– 0xFFF18A14	CLASS Initiator Profiling Configuration Register 5	C0IPCR5
– 0xFFF18A18	CLASS Initiator Profiling Configuration Register 6	C0IPCR6
– 0xFFF18A1C	CLASS Initiator Profiling Configuration Register 7	C0IPCR7
– 0xFFF18A20	CLASS Initiator Profiling Configuration Register 8	C0IPCR8
– 0xFFF18A24	CLASS Initiator Profiling Configuration Register 9	C0IPCR9
– 0xFFF18A28	CLASS Initiator Profiling Configuration Register 10	C0IPCR10
– 0xFFF18A2C	CLASS Initiator Profiling Configuration Register 11	C0IPCR11
– 0xFFF18A30–0xFFF18A3F	reserved	
– 0xFFF18A40	CLASS Initiator Watch Point Control Register 0	C0IWPCR0
– 0xFFF18A44	CLASS Initiator Watch Point Control Register 1	C0IWPCR1
– 0xFFF18A48	CLASS Initiator Watch Point Control Register 2	C0IWPCR2
– 0xFFF18A4C	CLASS Initiator Watch Point Control Register 3	C0IWPCR3
– 0xFFF18A50	CLASS Initiator Watch Point Control Register 4	C0IWPCR4
– 0xFFF18A54	CLASS Initiator Watch Point Control Register 5	C0IWPCR5
– 0xFFF18A58	CLASS Initiator Watch Point Control Register 6	C0IWPCR6
– 0xFFF18A5C	CLASS Initiator Watch Point Control Register 7	C0IWPCR7
– 0xFFF18A60	CLASS Initiator Watch Point Control Register 8	C0IWPCR8
– 0xFFF18A64	CLASS Initiator Watch Point Control Register 9	C0IWPCR9
– 0xFFF18A68	CLASS Initiator Watch Point Control Register 10	C0IWPCR10
– 0xFFF18A6C	CLASS Initiator Watch Point Control Register 11	C0IWPCR11
– 0xFFF18A70–0xFFF18A7F	reserved	
– 0xFFF18A80	CLASS Arbitration Weight Register 0	C0AWR0
– 0xFFF18A84	CLASS Arbitration Weight Register 1	C0AWR1
– 0xFFF18A88	CLASS Arbitration Weight Register 2	C0AWR2
– 0xFFF18A8C	CLASS Arbitration Weight Register 3	C0AWR3
– 0xFFF18A90	CLASS Arbitration Weight Register 4	C0AWR4
– 0xFFF18A94	CLASS Arbitration Weight Register 5	C0AWR5
– 0xFFF18A98	CLASS Arbitration Weight Register 6	C0AWR6

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF18A9C	CLASS Arbitration Weight Register 7	C0AWR7
– 0xFFF18AA0	CLASS Arbitration Weight Register 8	C0AWR8
– 0xFFF18AA4	CLASS Arbitration Weight Register 9	C0AWR9
– 0xFFF18AA8	CLASS Arbitration Weight Register 10	C0AWR10
– 0xFFF18AAC	CLASS Arbitration Weight Register 11	C0AWR11
– 0xFFF18AB0–0xFFF18C13	reserved	
– 0xFFF18C14	CLASS Start Address Decoder 5	C0SAD5
– 0xFFF18C18	CLASS Start Address Decoder 6	C0SAD6
– 0xFFF18C1C–0xFFF18C53	reserved	
– 0xFFF18C54	CLASS End Address Decoder 5	C0EAD5
– 0xFFF18C58	CLASS End Address Decoder 6	C0EAD6
– 0xFFF18C5C–0xFFF18C93	reserved	
– 0xFFF18C94	CLASS Attributes Decoder 5	C0ATD5
– 0xFFF18C98	CLASS Attributes Decoder 6	C0ATD6
– 0xFFF18C9C–0xFFF18D7F	reserved	
– 0xFFF18D80	CLASS IRQ Status Register	C0ISR
– 0xFFF18D84–0xFFF18DBF	Reserved	
– 0xFFF18DC0	CLASS IRQ Enable Register	C0IER
– 0xFFF18DC4–0xFFF18DFF	Reserved	
– 0xFFF18E00	CLASS Target Profiling Configuration Register	C0TPCR
– 0xFFF18E04	CLASS Profiling Control Register	C0PCR
– 0xFFF18E08	CLASS Watch Point Control Register	C0WPCR
– 0xFFF18E0C	CLASS Watch Point Access Configuration Register	C0WPACR
– 0xFFF18E10	CLASS Watch Point Extended Access Configuration Register	C0WPEACR
– 0xFFF18E14	CLASS Watch Point Address Mask Register	C0WPAMR
– 0xFFF18E18	CLASS Profiling Time-Out Register	C0PTOR
– 0xFFF18E1C	CLASS Target Watch Point Control Register	C0TWPCR
– 0xFFF18E20	CLASS Profiling IRQ Status Register	C0PISR
– 0xFFF18E24	CLASS Profiling IRQ Enable Register	C0PIER
– 0xFFF18E28-0xFFF18E3F	Reserved	
– 0xFFF18E40	CLASS Profiling Reference Counter Register	C0PRCR
– 0xFFF18E44	CLASS Profiling General Counter Register 0	C0PGCR0
– 0xFFF18E48	CLASS Profiling General Counter Register 1	C0PGCR1
– 0xFFF18E4C	CLASS Profiling General Counter Register 2	C0PGCR2
– 0xFFF18E50	CLASS Profiling General Counter Register 3	C0PGCR3
– 0xFFF18E54–0xFFF18FBF	Reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF18FC0	CLASS Arbitration Control Register	C0ACR
– 0xFFF18FC4–0xFFF18FFF	Reserved	
• 0xFFF19000–0xFFF1BFFF	reserved	
• 0xFFF1C000–0xFFF1FFFF	MAPLE-B (see **Chapter 26**, *Multi Accelerator Platform Engine, Baseband (MAPLE-B)*)	
– 0xFFF1C000–0xFFF1CFFF	reserved	
– 0xFFF1D000–0xFFF1D7FF	PSIF_PIC	
– 0xFFF1D600	PSIF PIC Event Register	PSPICER
– 0xFFF1D604	PSIF PIC Edge/Level Register	PSPICELR
– 0xFFF1D608	PSIF PIC Mask Register	PSPICMR
– 0xFFF1D60C	PSIF PIC Interrupt Assertion Clocks Register	PSPICIACR
– 0xFFF1D610–0xFFF1D7FF	reserved	
– 0xFFF1D800–0xFFF1FFFF	reserved	
• 0xFFF20000–0xFFF21FFF	DDR Controller 1 (see **Chapter 12**, *DDR SDRAM Memory Controller*)	
– 0xFFF20000	Chip Select 0 Bounds	M1CS0_BNDS
– 0xFFF20004–0xFFF20007	reserved	
– 0xFFF20008	Chip Select 1 Bounds	M1CS1_BNDS
– 0xFFF2000C–0xFFF2007F	reserved	
– 0xFFF20080	Chip Select 0 Configuration	M1CS0_CONFIG
– 0xFFF20084	Chip Select 1 Configuration	M1CS1_CONFIG
– 0xFFF20088–0xFFF200BF	reserved	
– 0xFFF200C0	Chip Select 0 Configuration 2	M1CS0_CONFIG_2
– 0xFFF200C4	Chip Select 1 Configuration 2	M1CS1_CONFIG_2
– 0xFFF200C8–0xFFF200FF	reserved	
– 0xFFF20100	DDR SDRAM Timing Configuration 3	M1TIMING_CFG_3
– 0xFFF20104	DDR SDRAM Timing Configuration 0	M1TIMING_CFG_0
– 0xFFF20108	DDR SDRAM Timing Configuration 1	M1TIMING_CFG_1
– 0xFFF2010C	DDR SCRAM Timing Configuration 2	M1TIMING_CFG_2
– 0xFFF20110	DDR SDRAM Control Configuration	M1DDR_SDRAM_CFG
– 0xFFF20114	DDR SDRAM Control Configuration 2	M1DDR_SDRAM_CFG_2
– 0xFFF20118	DDR SDRAM Mode Configuration	M1DDR_SDRAM_MODE

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF2011C	DDR SDRAM Mode Configuration 2	M1DDR_SDRAM_MODE_2
– 0xFFF20120	DDR SDRAM Mode Control	M1DDR_SCRAM_MD_CNTL
– 0xFFF20124	DDR SDRAM Interval Configuration	M1DDR_SDRAM_INTERVAL
– 0xFFF20128	DDR SDRAM Data Initialization	M1DDR_DATA_INIT
– 0xFFF2012C–0xFFF2012F	reserved	
– 0xFFF20130	DDR SDRAM Clock Control	M1DDR_SDRAM_CLK_CNT
– 0xFFF20134–0xFFF20147	reserved	
– 0xFFF20148	DDR SDRAM Initialization Address	M1DDR_INIT_ADDRESS
– 0xFFF2014C	DDR Initialization Enable Extended Address	M1DDR_INIT_ENXT_ADDR
– 0xFFF20150–0xFFF2015F	reserved	
– 0xFFF20160	DDR SDRAM Timing Configuration 4	M1TIMING_CFG_4
– 0xFFF20164	DDR SDRAM Timing Configuration 5	M1TIMING_CFG_5
– 0xFFF20168–0xFFF2016F	reserved	
– 0xFFF20170	DDR ZQ Calibration Control	M1DDR_ZQ_CNTL
– 0xFFF20174	DDR Write Leveling Control	M1DDR_WRLVL_CNTL
– 0xFFF22178	DDR Pre-Drive Conditioning Control	M1DDR_PD_CNTL
– 0xFFF2017C	DDR Self Refresh Counter	M1DDR_SR_CNTR
– 0xFFF20180	DDR SDRAM Register Control Words 1	M1DDR_SDRAM_RCW_1
– 0xFFF20184	DDR SDRAM Register Control Words 2	M1DDR_SDRAM_RCW_2
– 0xFFF20188–0xFFF2018F	reserved	
– 0xFFF20190	DDR Write Leveling Control 2	M1DDR_WRLVL_CNTL_2
– 0xFFF20194	DDR Write Leveling Control 3	M1DDR_WRLVL_CNTL_3
– 0xFFF20198–0xFFF20B1F	reserved	
– 0xFFF20B20	DDR Debug Status Register 1	M1DDRDSR_1
– 0xFFF20B24	DDR Debug Status Register 2	M1DDRDSR_2
– 0xFFF20B28	DDR Control Driver Register 1	M1DDRCDR_1
– 0xFFF20B2C	DDR Control Driver Register 2	M1DDRCDR_2
– 0xFFF20B30–0xFFF20BF7	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF20BF8	DDR IP Block Revision 1	M1DDR_IP_REV1
– 0xFFF20BFC	DDR IP Block Revision 2	M1DDR_IP_REV2
– 0xFFF20C00–0xFFF20DFF	reserved	
– 0xFFF20E00	Memory Data Path Error Injection Mask High	M1DDR_ERR_INJECT_HI
– 0xFFF20E04	Memory Data Path Error Injection Mask Low	M1DDR_ERR_INJECT_LO
– 0xFFF20E08	Memory Data Path Error Injection Mask ECC	M1DDR_ERR_INJECT
– 0xFFF20E0C–0xFFF20E1F	reserved	
– 0xFFF20E20	Memory Data Path Read Capture High	M1CAPTURE_DATA_HI
– 0xFFF20E24	Memory Data Path Read Capture Low	M1CAPTURE-DATA_LO
– 0xFFF20E28	Memory Data Path Read Capture ECC	M1CAPTURE_ECC
– 0xFFF20E2C–0xFFF20E3F	reserved	
– 0xFFF20E40	Memory Error Detect	M1ERR_DETECT
– 0xFFF20E44	Memory Error Disable	M1ERR_DISABLE
– 0xFFF20E48	Memory Error Interrupt Enable	M1ERR_INT_EN
– 0xFFF20E4C	Memory Error Attributes Capture	M1CAPTURE_ATTRIBUTES
– 0xFFF20E50	Memory Error Address Capture	M1CAPTURE_ADDRESS
– 0xFFF20E54–0xFFF20E57	reserved	
– 0xFFF20E58	Single-Bit ECC Memory Error Management	M1ERR_SBE
– 0xFFF20E5C–0xFFF20F03	reserved	
– 0xFFF20F04	Debug Register 2	M1DEBUG_2
– 0xFFF20F08–0xFFF21FFF	reserved	
• 0xFFF22000–0xFFF23FFF	DDR Controller 2 (see **Chapter 12**, *DDR SDRAM Memory Controller*)	
– 0xFFF22000	Chip Select 0 Memory Bounds	M2CS0_BNDS
– 0xFFF22004–0xFFF22007	reserved	
– 0xFFF22008	Chip Select 1 Memory Bounds	M2CS1_BNDS
– 0xFFF2200C–0xFFF2207F	reserved	
– 0xFFF22080	Chip Select 0 Configuration	M2CS0_CONFIG
– 0xFFF22084	Chip Select 1 Configuration	M2CS1_CONFIG
– 0xFFF22088–0xFFF220BF	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF220C0	Chip Select 0 Configuration 2	M2CS0_CONFIG_2
– 0xFFF220C4	Chip Select 1 Configuration 2	M2CS1_CONFIG_2
– 0xFFF220C8–0xFFF220FF	reserved	
– 0xFFF22100	DDR SDRAM Timing Configuration 3	M2TIMING_CFG_3
– 0xFFF22104	DDR SDRAM Timing Configuration 0	M2TIMING_CFG_0
– 0xFFF22108	DDR SDRAM Timing Configuration 1	M2TIMING_CFG_1
– 0xFFF2210C	DDR SCRAM Timing Configuration 2	M2TIMING_CFG_2
– 0xFFF22110	DDR SDRAM Control Configuration	M2DDR_SDRAM_CFG
– 0xFFF22114	DDR SDRAM Control Configuration 2	M2DDR_SDRAM_CFG_2
– 0xFFF22118	DDR SDRAM Mode Configuration	M2DDR_SDRAM_MODE
– 0xFFF2211C	DDR SDRAM Mode Configuration 2	M2DDR_SDRAM_MODE_2
– 0xFFF22120	DDR SDRAM Mode Control	M2DDR_SCRAM_MD_CNTL
– 0xFFF22124	DDR SDRAM Interval Configuration	M2DDR_SDRAM_INTERVAL
– 0xFFF22128	DDR SDRAM Data Initialization	M2DDR_DATA_INIT
– 0xFFF2212C–0xFFF2212F	reserved	
– 0xFFF22130	DDR SDRAM Clock Control	M2DDR_SDRAM_CLK_CNT
– 0xFFF22134–0xFFF22147	reserved	
– 0xFFF22148	DDR SDRAM Initialization Address	M2DDR_INIT_ADDRESS
– 0xFFF2214C	DDR Initialization Enable Extended Address	M2DDR_INIT_ENXT_ADDR
– 0xFFF22150–0xFFF2215F	reserved	
– 0xFFF22160	DDR SDRAM Timing Configuration 4	M2TIMING_CFG_4
– 0xFFF22164	DDR SDRAM Timing Configuration 5	M2TIMING_CFG_5
– 0xFFF22168–0xFFF2216F	reserved	
– 0xFFF22170	DDR ZQ Calibration Control	M2DDR_ZQ_CNTL
– 0xFFF22174	DDR Write Leveling Control	M2DDR_WRLVL_CNTL
– 0xFFF22178	DDR Pre-Drive Conditioning Control	M2DDR_PD_CNTL
– 0xFFF2217C	DDR Self Refresh Counter	M2DDR_SR_CNTR
– 0xFFF22180	DDR SDRAM Register Control Words 1	M2DDR_SDRAM_RCW_1
– 0xFFF22184	DDR SDRAM Register Control Words 2	M2DDR_SDRAM_RCW_2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF22188–0xFFF2218F	reserved	
– 0xFFF22190	DDR Write Leveling Control 2	M2DDR_WRLVL_CNTL_2
– 0xFFF22194	DDR Write Leveling Control 3	M2DDR_WRLVL_CNTL_3
– 0xFFF22198–0xFFF22B1F	reserved	
– 0xFFF22B20	DDR Debug Status Register 1	M2DDRDSR_1
– 0xFFF22B24	DDR Debug Status Register 2	M2DDRDSR_2
– 0xFFF22B28	DDR Control Driver Register 1	M2DDRCDR_1
– 0xFFF22B2C	DDR Control Driver Register 2	M2DDRCDR_2
– 0xFFF22B30–0xFFF22BF7	reserved	
– 0xFFF22BF8	DDR IP Block Revision 1	M2DDR_IP_REV1
– 0xFFF22BFC	DDR IP Block Revision 2	M2DDR_IP_REV2
– 0xFFF22C00–0xFFF22DFF	reserved	
– 0xFFF22E00	Memory Data Path Error Injection Mask High	M2DDR_ERR_INJECT_HI
– 0xFFF22E04	Memory Data Path Error Injection Mask Low	M2DDR_ERR_INJECT_LO
– 0xFFF22E08	Memory Data Path Error Injection Mask ECC	M2DDR_ERR_INJECT
– 0xFFF22E0C–0xFFF22E1F	reserved	
– 0xFFF22E20	Memory Data Path Read Capture High	M2CAPTURE_DATA_HI
– 0xFFF22E24	Memory Data Path Read Capture Low	M2CAPTURE-DATA_LO
– 0xFFF22E28	Memory Data Path Read Capture ECC	M2CAPTURE_ECC
– 0xFFF22E2C–0xFFF22E3F	reserved	
– 0xFFF22E40	Memory Error Detect	M2ERR_DETECT
– 0xFFF22E44	Memory Error Disable	M2ERR_DISABLE
– 0xFFF22E48	Memory Error Interrupt Enable	M2ERR_INT_EN
– 0xFFF22E4C	Memory Error Attributes Capture	M2CAPTURE_ATTRIBUTES
– 0xFFF22E50	Memory Error Address Capture	M2CAPTURE_ADDRESS
– 0xFFF22E54–0xFFF22E57	reserved	
– 0xFFF22E58	Single-Bit ECC Memory Error Management	M2ERR_SBE
– 0xFFF22E5C–0xFFF22F03	reserved	
– 0xFFF22F04	Debug Register 2	M2DEBUG_2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF22F08–0xFFF23FFF	reserved	
• 0xFFF24000–0xFFF2407F	Clocks (see **Chapter 7**, *Clocks*)	
– 0xFFF24000	System Clock Control Register	SCCR
– 0xFFF24004	Clock General Purpose Register 0	CLK_GPR0
– 0xFFF24008–0xFFF2407F	reserved	
• 0xFFF24080–0xFFF247FF	reserved	
• 0xFFF24800–0xFFF248FF	Reset (see **Chapter 5**, *Reset*)	
– 0xFFF24800	Reset Configuration Word Low Register	RCWLR
– 0xFFF24804	Reset Configuration Word High Register	RCWHR
– 0xFFF24808–0xFFF2480F	reserved	
– 0xFFF24810	Reset Status Register	RSR
– 0xFFF24814–0xFFF24817	reserved	
– 0xFFF24818	Reset Protection Register	RPR
– 0xFFF2481C	Reset Control Register	RCR
– 0xFFF24820	Reset Control Enable Register	RCER
– 0xFFF24824–0xFFF248FF	reserved	
• 0xFFF24900–0xFFF24BFF	reserved	
• 0xFFF24C00–0xFFF24CFF	I^2C (see **Chapter 24**, *I2C*)	
– 0xFFF24C00	I^2C Address Register	I2CADR
– 0xFFF24C04	I^2C Frequency Divider Register	I2CFDR
– 0xFFF24C08	I^2C Control Register	I2CCR
– 0xFFF24C0C	I^2C Status Register	I2CSR
– 0xFFF24C10	I^2C Data Register	I2CDR
– 0xFFF24C14	I^2C Digital Filter Sampling Rate Register	I2CDFSRR
– 0xFFF24C18–0xFFF24CFF	reserved	
• 0xFFF24D00–0xFFF24FFF	reserved	
• 0xFFF25000–0xFFF250FF	Watchdog Timer 0 (see **Chapter 21**, *Timers*)	
– 0xFFF25000–0xFFF25003	reserved	
– 0xFFF25004	System Watchdog Control Register 0	SWCRR0
– 0xFFF25008	System Watchdog Count Register 0	SWCNR0
– 0xFFF2500C–0xFFF2500D	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF2500E	System Watchdog Service Register 0	SWSRR0
– 0xFFF25010–0xFFF250FF	reserved	
• 0xFFF25100–0xFFF251FF	Watchdog Timer 1 (see **Chapter 21**, *Timers*)	
– 0xFFF25100–0xFFF25103	reserved	
– 0xFFF25104	System Watchdog Control Register 1	SWCRR1
– 0xFFF25108	System Watchdog Count Register 1	SWCNR1
– 0xFFF2510C–0xFFF2510D	reserved	
– 0xFFF2510E	System Watchdog Service Register 1	SWSRR1
– 0xFFF25110–0xFFF251FF	reserved	
• 0xFFF25200–0xFFF252FF	Watchdog Timer 2 (see **Chapter 21**, *Timers*)	
– 0xFFF25200–0xFFF25203	reserved	
– 0xFFF25204	System Watchdog Control Register 2	SWCRR2
– 0xFFF25208	System Watchdog Count Register 2	SWCNR2
– 0xFFF2520C–0xFFF2520D	reserved	
– 0xFFF2520E	System Watchdog Service Register 2	SWSRR2
– 0xFFF25210–0xFFF252FF	reserved	
• 0xFFF25300–0xFFF253FF	Watchdog Timer 3 (see **Chapter 21**, *Timers*)	
– 0xFFF25300–0xFFF25303	reserved	
– 0xFFF25304	System Watchdog Control Register 3	SWCRR3
– 0xFFF25308	System Watchdog Count Register 3	SWCNR3
– 0xFFF2530C–0xFFF2530D	reserved	
– 0xFFF2530E	System Watchdog Service Register 3	SWSRR3
– 0xFFF25310–0xFFF253FF	reserved	
• 0xFFF25400–0xFFF254FF	Watchdog Timer 4 (see **Chapter 21**, *Timers*)	
– 0xFFF25400–0xFFF25403	reserved	
– 0xFFF25404	System Watchdog Control Register 4	SWCRR4
– 0xFFF25408	System Watchdog Count Register 4	SWCNR4
– 0xFFF2540C–0xFFF2540D	reserved	
– 0xFFF2540E	System Watchdog Service Register 4	SWSRR4

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF25410–0xFFF254FF	reserved	
• 0xFFF25500–0xFFF255FF	Watchdog Timer 5 (see **Chapter 21**, *Timers*)	
– 0xFFF25500–0xFFF25503	reserved	
– 0xFFF25504	System Watchdog Control Register 5	SWCRR5
– 0xFFF25508	System Watchdog Count Register 5	SWCNR5
– 0xFFF2550C–0xFFF2550D	reserved	
– 0xFFF2550E	System Watchdog Service Register 5	SWSRR5
– 0xFFF25510–0xFFF255FF	reserved	
• 0xFFF25600–0xFFF256FF	Watchdog Timer 6 (see **Chapter 21**, *Timers*)	
– 0xFFF25600–0xFFF25603	reserved	
– 0xFFF25604	System Watchdog Control Register 6	SWCRR6
– 0xFFF25608	System Watchdog Count Register 6	SWCNR6
– 0xFFF2560C–0xFFF2560D	reserved	
– 0xFFF2560E	System Watchdog Service Register 6	SWSRR6
– 0xFFF25610–0xFFF256FF	reserved	
• 0xFFF25700–0xFFF257FF	Watchdog Timer 7 (see **Chapter 21**, *Timers*)	
– 0xFFF25700–0xFFF25703	reserved	
– 0xFFF25704	System Watchdog Control Register 7	SWCRR7
– 0xFFF25708	System Watchdog Count Register 7	SWCNR7
– 0xFFF2570C–0xFFF2570D	reserved	
– 0xFFF2570E	System Watchdog Service Register 7	SWSRR7
– 0xFFF25710–0xFFF257FF	reserved	
• 0xFFF25800–0xFFF25FFF	reserved	
• 0xFFF26000–0xFFF260FF	Timer 0 (see **Chapter 21**, *Timers*)	
– 0xFFF26000	Timer 0 Channel 0 Compare 1 Register	TMR0CMP10
– 0xFFF26004	Timer 0 Channel 0 Compare 2 Register	TMR0CMP20
– 0xFFF26008	Timer 0 Channel 0 Capture Register	TMR0CAP0
– 0xFFF2600C	Timer 0 Channel 0 Load Register	TMR0LD0
– 0xFFF26010	Timer 0 Channel 0 Hold Register	TMR0HOLD0
– 0xFFF26014	Timer 0 Channel 0 Counter Register	TMR0CNTR0
– 0xFFF26018	Timer 0 Channel 0 Control Register	TMR0CTL0

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF2601C	Timer 0 Channel 0 Status and Control Register	TMR0SCTL0
– 0xFFF26020	Timer 0 Channel 0 Compare Load 1 Register	TMR0CMPLD10
– 0xFFF26024	Timer 0 Channel 0 Compare Load 2 Register	TMR0CMPLD20
– 0xFFF26028	Timer 0 Channel 0 Comparator Status and Control Register	TMR0COMSC0
– 0xFFF2602C–0xFFF2603F	reserved	
– 0xFFF26040	Timer 0 Channel 1 Compare 1 Register	TMR0CMP11
– 0xFFF26044	Timer 0 Channel 1 Compare 2 Register	TMR0CMP21
– 0xFFF26048	Timer 0 Channel 1 Capture Register	TMR0CAP1
– 0xFFF2604C	Timer 0 Channel 1 Load Register	TMR0LOAD1
– 0xFFF26050	Timer 0 Channel 1 Hold Register	TMR0HOLD1
– 0xFFF26054	Timer 0 Channel 1 Counter Register	TMR0CNTR1
– 0xFFF26058	Timer 0 Channel 1 Control Register	TMR0CTL1
– 0xFFF2605C	Timer 0 Channel 1 Status and Control Register	TMR0SCTL1
– 0xFFF26060	Timer 0 Channel 1 Compare Load 1 Register	TMR0CMPLD11
– 0xFFF26064	Timer 0 Channel 1 Compare Load 2 Register	TMR0CMPLD21
– 0xFFF26068	Timer 0 Channel 1 Comparator Status and Control Register	TMR0COMSC1
– 0xFFF2606C–0xFFF2607F	reserved	
– 0xFFF26080	Timer 0 Channel 2 Compare 1 Register	TMR0CMP12
– 0xFFF26084	Timer 0 Channel 2 Compare 2 Register	TMR0CMP22
– 0xFFF26088	Timer 0 Channel 2 Capture Register	TMR0CAP2
– 0xFFF2608C	Timer 0 Channel 2 Load Register	TMR0LOAD2
– 0xFFF26090	Timer 0 Channel 2 Hold Register	TMR0HOLD2
– 0xFFF26094	Timer 0 Channel 2 Counter Register	TMR0CNTR2
– 0xFFF26098	Timer 0 Channel 2 Control Register	TMR0CTL2
– 0xFFF2609C	Timer 0 Channel 2 Status and Control Register	TMR0SCTL2
– 0xFFF260A0	Timer 0 Channel 2 Compare Load 1 Register	TMR0CMPLD12
– 0xFFF260A4	Timer 0 Channel 2 Compare Load 2 Register	TMR0CMPLD22
– 0xFFF260A8	Timer 0 Channel 2 Comparator Status and Control Register	TMR0COMSC2
– 0xFFF260AC–0xFFF260BF	reserved	
– 0xFFF260C0	Timer 0 Channel 3 Compare 1 Register	TMR0CMP13
– 0xFFF260C4	Timer 0 Channel 3 Compare 2 Register	TMR0CMP23
– 0xFFF260C8	Timer 0 Channel 3 Capture Register	TMR0CAP3
– 0xFFF260CC	Timer 0 Channel 3 Load Register	TMR0LOAD3
– 0xFFF260D0	Timer 0 Channel 3 Hold Register	TMR0HOLD3
– 0xFFF260D4	Timer 0 Channel 3 Counter Register	TMR0CNTR3
– 0xFFF260D8	Timer 0 Channel 3 Control Register	TMR0CTL3
– 0xFFF260DC	Timer 0 Channel 3 Status and Control Register	TMR0SCTL3
– 0xFFF260E0	Timer 0 Channel 3 Compare Load 1 Register	TMR0CMPLD13
– 0xFFF260E4	Timer 0 Channel 3 Compare Load 2 Register	TMR0CMPLD23

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF260E8	Timer 0 Channel 3 Comparator Status and Control Register	TMR0COMSC3
– 0xFFF260EC–0xFFF260FF	reserved	
• 0xFFF26100–0xFFF261FF	Timer 1 (see **Chapter 21**, *Timers*)	
– 0xFFF26100	Timer 1 Channel 0 Compare 1 Register	TMR1CMP10
– 0xFFF26104	Timer 1 Channel 0 Compare 2 Register	TMR1CMP20
– 0xFFF26108	Timer 1 Channel 0 Capture Register	TMR1CAP0
– 0xFFF2610C	Timer 1 Channel 0 Load Register	TMR1LOAD0
– 0xFFF26110	Timer 1 Channel 0 Hold Register	TMR1HOLD0
– 0xFFF26114	Timer 1 Channel 0 Counter Register	TMR1CNTR0
– 0xFFF26118	Timer 1 Channel 0 Control Register	TMR1CTL0
– 0xFFF2611C	Timer 1 Channel 0 Status and Control Register	TMR1SCTL0
– 0xFFF26120	Timer 1 Channel 0 Compare Load 1 Register	TMR1CMPLD10
– 0xFFF26124	Timer 1 Channel 0 Compare Load 2 Register	TMR1CMPLD20
– 0xFFF26128	Timer 1 Channel 0 Comparator Status and Control Register	TMR1COMSC0
– 0xFFF2612C–0xFFF2613F	reserved	
– 0xFFF26140	Timer 1 Channel 1 Compare 1 Register	TMR1CMP11
– 0xFFF26144	Timer 1 Channel 1 Compare 2 Register	TMR1CMP21
– 0xFFF26148	Timer 1 Channel 1 Capture Register	TMR1CAP1
– 0xFFF2614C	Timer 1 Channel 1 Load Register	TMR1LOAD1
– 0xFFF26150	Timer 1 Channel 1 Hold Register	TMR1HOLD1
– 0xFFF26154	Timer 1 Channel 1 Counter Register	TMR1CNTR1
– 0xFFF26158	Timer 1 Channel 1 Control Register	TMR1CTL1
– 0xFFF2615C	Timer 1 Channel 1 Status and Control Register	TMR1SCTL1
– 0xFFF26160	Timer 1 Channel 1 Compare Load 1 Register	TMR1CMPLD11
– 0xFFF26164	Timer 1 Channel 1 Compare Load 2 Register	TMR1CMPLD21
– 0xFFF26168	Timer 1 Channel 1 Comparator Status and Control Register	TMR1COMSC1
– 0xFFF2616C–0xFFF2617F	reserved	
– 0xFFF26180	Timer 1 Channel 2 Compare 1 Register	TMR1CMP12
– 0xFFF26184	Timer 1 Channel 2 Compare 2 Register	TMR1CMP22
– 0xFFF26188	Timer 1 Channel 2 Capture Register	TMR1CAP2
– 0xFFF2618C	Timer 1 Channel 2 Load Register	TMR1LOAD2
– 0xFFF26190	Timer 1 Channel 2 Hold Register	TMR1HOLD2
– 0xFFF26194	Timer 1 Channel 2 Counter Register	TMR1CNTR2
– 0xFFF26198	Timer 1 Channel 2 Control Register	TMR1CTL2
– 0xFFF2619C	Timer 1 Channel 2 Status and Control Register	TMR1SCTL2
– 0xFFF261A0	Timer 1 Channel 2 Compare Load 1 Register	TMR1CMPLD12
– 0xFFF261A4	Timer 1 Channel 2 Compare Load 2 Register	TMR1CMPLD22
– 0xFFF261A8	Timer 1 Channel 2 Comparator Status and Control Register	TMR1COMSC2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF261AC–0xFFF261BF	reserved	
– 0xFFF261C0	Timer 1 Channel 3 Compare 1 Register	TMR1CMP13
– 0xFFF261C4	Timer 1 Channel 3 Compare 2 Register	TMR1CMP23
– 0xFFF261C8	Timer 1 Channel 3 Capture Register	TMR1CAP3
– 0xFFF261CC	Timer 1 Channel 3 Load Register	TMR1LOAD3
– 0xFFF261D0	Timer 1 Channel 3 Hold Register	TMR1HOLD3
– 0xFFF261D4	Timer 1 Channel 3 Counter Register	TMR1CNTR3
– 0xFFF261D8	Timer 1 Channel 3 Control Register	TMR1CTL3
– 0xFFF261DC	Timer 1 Channel 3 Status and Control Register	TMR1SCTL3
– 0xFFF261E0	Timer 1 Channel 3 Compare Load 1 Register	TMR1CMPLD13
– 0xFFF261E4	Timer 1 Channel 3 Compare Load 2 Register	TMR1CMPLD23
– 0xFFF261E8	Timer 1 Channel 3 Comparator Status and Control Register	TMR1COMSC3
– 0xFFF261EC–0xFFF261FF	reserved	
• 0xFFF26200–0xFFF262FF	Timer 2 (see **Chapter 21**, *Timers*)	
– 0xFFF26200	Timer 2 Channel 0 Compare 1 Register	TMR2CMP10
– 0xFFF26204	Timer 2 Channel 0 Compare 2 Register	TMR2CMP20
– 0xFFF26208	Timer 2 Channel 0 Capture Register	TMR2CAP0
– 0xFFF2620C	Timer 2 Channel 0 Load Register	TMR2LOAD0
– 0xFFF26210	Timer 2 Channel 0 Hold Register	TMR2HOLD0
– 0xFFF26214	Timer 2 Channel 0 Counter Register	TMR2CNTR0
– 0xFFF26218	Timer 2 Channel 0 Control Register	TMR2CTL0
– 0xFFF2621C	Timer 2 Channel 0 Status and Control Register	TMR2SCTL0
– 0xFFF26220	Timer 2 Channel 0 Compare Load 1 Register	TMR2CMPLD10
– 0xFFF26224	Timer 2 Channel 0 Compare Load 2 Register	TMR2CMPLD20
– 0xFFF26228	Timer 2 Channel 0 Comparator Status and Control Register	TMR2COMSC0
– 0xFFF2622C–0xFFF2623F	reserved	
– 0xFFF26240	Timer 2 Channel 1 Compare 1 Register	TMR2CMP11
– 0xFFF26244	Timer 2 Channel 1 Compare 2 Register	TMR2CMP21
– 0xFFF26248	Timer 2 Channel 1 Capture Register	TMR2CAP1
– 0xFFF2624C	Timer 2 Channel 1 Load Register	TMR2LOAD1
– 0xFFF26250	Timer 2 Channel 1 Hold Register	TMR2HOLD1
– 0xFFF26254	Timer 2 Channel 1 Counter Register	TMR2CNTR1
– 0xFFF26258	Timer 2 Channel 1 Control Register	TMR2CTL1
– 0xFFF2625C	Timer 2 Channel 1 Status and Control Register	TMR2SCTL1
– 0xFFF26260	Timer 2 Channel 1 Compare Load 1 Register	TMR2CMPLD11
– 0xFFF26264	Timer 2 Channel 1 Compare Load 2 Register	TMR2CMPLD21
– 0xFFF26268	Timer 2 Channel 1 Comparator Status and Control Register	TMR2COMSC1
– 0xFFF2626C–0xFFF2627F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF26280	Timer 2 Channel 2 Compare 1 Register	TMR2CMP12
– 0xFFF26284	Timer 2 Channel 2 Compare 2 Register	TMR2CMP22
– 0xFFF26288	Timer 2 Channel 2 Capture Register	TMR2CAP2
– 0xFFF2628C	Timer 2 Channel 2 Load Register	TMR2LOAD2
– 0xFFF26290	Timer 2 Channel 2 Hold Register	TMR2HOLD2
– 0xFFF26294	Timer 2 Channel 2 Counter Register	TMR2CNTR2
– 0xFFF26298	Timer 2 Channel 2 Control Register	TMR2CTL2
– 0xFFF2629C	Timer 2 Channel 2 Status and Control Register	TMR2SCTL2
– 0xFFF262A0	Timer 2 Channel 2 Compare Load 1 Register	TMR2CMPLD12
– 0xFFF262A4	Timer 2 Channel 2 Compare Load 2 Register	TMR2CMPLD22
– 0xFFF262A8	Timer 2 Channel 2 Comparator Status and Control Register	TMR2COMSC2
– 0xFFF262AC–0xFFF262BF	reserved	
– 0xFFF262C0	Timer 2 Channel 3 Compare 1 Register	TMR2CMP13
– 0xFFF262C4	Timer 2 Channel 3 Compare 2 Register	TMR2CMP23
– 0xFFF262C8	Timer 2 Channel 3 Capture Register	TMR2CAP3
– 0xFFF262CC	Timer 2 Channel 3 Load Register	TMR2LOAD3
– 0xFFF262D0	Timer 2 Channel 3 Hold Register	TMR2HOLD3
– 0xFFF262D4	Timer 2 Channel 3 Counter Register	TMR2CNTR3
– 0xFFF262D8	Timer 2 Channel 3 Control Register	TMR2CTL3
– 0xFFF262DC	Timer 2 Channel 3 Status and Control Register	TMR2SCTL3
– 0xFFF262E0	Timer 2 Channel 3 Compare Load 1 Register	TMR2CMPLD13
– 0xFFF262E4	Timer 2 Channel 3 Compare Load 2 Register	TMR2CMPLD23
– 0xFFF262E8	Timer 2 Channel 3 Comparator Status and Control Register	TMR2COMSC3
– 0xFFF262EC–0xFFF262FF	reserved	
• 0xFFF26300–0xFFF263FF	Timer 3 (see **Chapter 21**, *Timers*)	
– 0xFFF26300	Timer 3 Channel 0 Compare 1 Register	TMR3CMP10
– 0xFFF26304	Timer 3 Channel 0 Compare 2 Register	TMR3CMP20
– 0xFFF26308	Timer 3 Channel 0 Capture Register	TMR3CAP0
– 0xFFF2630C	Timer 3 Channel 0 Load Register	TMR3LOAD0
– 0xFFF26310	Timer 3 Channel 0 Hold Register	TMR3HOLD0
– 0xFFF26314	Timer 3 Channel 0 Counter Register	TMR3CNTR0
– 0xFFF26318	Timer 3 Channel 0 Control Register	TMR3CTL0
– 0xFFF2631C	Timer 3 Channel 0 Status and Control Register	TMR3SCTL0
– 0xFFF26320	Timer 3 Channel 0 Load 1 Register	TMR3CMPLD10
– 0xFFF26324	Timer 3 Channel 0 Load 2 Register	TMR3CMPLD20
– 0xFFF26328	Timer 3 Channel 0 Comparator Status and Control Register	TMR3COMSC0
– 0xFFF2632C–0xFFF2633F	reserved	
– 0xFFF26340	Timer 3 Channel 1 Compare 1 Register	TMR3CMP11
– 0xFFF26344	Timer 3 Channel 1 Compare 2 Register	TMR3CMP21

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF26348	Timer 3 Channel 1 Capture Register	TMR3CAP1
– 0xFFF2634C	Timer 3 Channel 1 Load Register	TMR3LOAD1
– 0xFFF26350	Timer 3 Channel 1 Hold Register	TMR3HOLD1
– 0xFFF26354	Timer 3 Channel 1 Counter Register	TMR3CNTR1
– 0xFFF26358	Timer 3 Channel 1 Control Register	TMR3CTL1
– 0xFFF2635C	Timer 3 Channel 1 Status and Control Register	TMR3SCTL1
– 0xFFF26360	Timer 3 Channel 1 Load 1 Register	TMR3CMPLD11
– 0xFFF26364	Timer 3 Channel 1 Load 2 Register	TMR3CMPLD21
– 0xFFF26368	Timer 3 Channel 1 Comparator Status and Control Register	TMR3COMSC1
– 0xFFF2636C–0xFFF2637F	reserved	
– 0xFFF26380	Timer 3 Channel 2 Compare 1 Register	TMR3CMP12
– 0xFFF26384	Timer 3 Channel 2 Compare 2 Register	TMR3CMP22
– 0xFFF26388	Timer 3 Channel 2 Capture Register	TMR3CAP2
– 0xFFF2638C	Timer 3 Channel 2 Load Register	TMR3LOAD2
– 0xFFF26390	Timer 3 Channel 2 Hold Register	TMR3HOLD2
– 0xFFF26394	Timer 3 Channel 2 Counter Register	TMR3CNTR2
– 0xFFF26398	Timer 3 Channel 2 Control Register	TMR3CTL2
– 0xFFF2639C	Timer 3 Channel 2 Status and Control Register	TMR3SCTL2
– 0xFFF263A0	Timer 3 Channel 2 Load 1 Register	TMR3CMPLD12
– 0xFFF263A4	Timer 3 Channel 2 Load 2 Register	TMR3CMPLD22
– 0xFFF263A8	Timer 3 Channel 2 Comparator Status and Control Register	TMR3COMSC2
– 0xFFF263AC–0xFFF263BF	reserved	
– 0xFFF263C0	Timer 3 Channel 3 Compare 1 Register	TMR3CMP13
– 0xFFF263C4	Timer 3 Channel 3 Compare 2 Register	TMR3CMP23
– 0xFFF263C8	Timer 3 Channel 3 Capture Register	TMR3CAP3
– 0xFFF263CC	Timer 3 Channel 3 Load Register	TMR3LOAD3
– 0xFFF263D0	Timer 3 Channel 3 Hold Register	TMR3HOLD3
– 0xFFF263D4	Timer 3 Channel 3 Counter Register	TMR3CNTR3
– 0xFFF263D8	Timer 3 Channel 3 Control Register	TMR3CTL3
– 0xFFF263DC	Timer 3 Channel 3 Status and Control Register	TMR3SCTL3
– 0xFFF263E0	Timer 3 Channel 3 Load 1 Register	TMR3CMPLD13
– 0xFFF263E4	Timer 3 Channel 3 Load 2 Register	TMR3CMPLD23
– 0xFFF263E8	Timer 3 Channel 3 Comparator Status and Control Register	TMR3COMSC3
– 0xFFF263EC–0xFFF263FF	reserved	
• 0xFFF26400–0xFFF26BFF	reserved	
• 0xFFF26C00–0xFFF26C3F	UART (see **Chapter 20**, *UART*)	
– 0xFFF26C00	SCI Baud-Rate Register	SCIBR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF26C04–0xFFF26C07	reserved	
– 0xFFF26C08	SCI Control Register	SCICR
– 0xFFF26C0C–0xFFF26C0F	reserved	
– 0xFFF26C10	SCI Status Register	SCISR
– 0xFFF26C14–0xFFF26C17	reserved	
– 0xFFF26C18	SCI Data Register	SCIDR
– 0xFFF26C1C–0xFFF26C27	reserved	
– 0xFFF26C28	SCI Data Direction Register	SCIDDR
– 0xFFF26C2C–0xFFF26C3F	reserved	
• 0xFFF26C40–0xFFF26FFF	reserved	
• 0xFFF27000–0xFFF270FF	GIC (see **Chapter 13**, *Interrupt Handling*)	
– 0xFFF27000	Virtual Interrupt Generation Register	VIGR
– 0xFFF27004–0xFFF27007	reserved	
– 0xFFF27008	Virtual Interrupt Status Register	VISR
– 0xFFF2700C–0xFFF270FF	reserved	
• 0xFFF27100–0xFFF271FF	Hardware Semaphores (see **Chapter 23**, *Hardware Semaphores*)	
– 0xFFF27100	Hardware Semaphore Register 0	HSMPR0
– 0xFFF27104–0xFFF27107	reserved	
– 0xFFF27108	Hardware Semaphore Register 1	HSMPR1
– 0xFFF2710C–0xFFF2710F	reserved	
– 0xFFF27110	Hardware Semaphore Register 2	HSMPR2
– 0xFFF27114–0xFFF27117	reserved	
– 0xFFF27118	Hardware Semaphore Register 3	HSMPR3
– 0xFFF2711C–0xFFF2711F	reserved	
– 0xFFF27120	Hardware Semaphore Register 4	HSMPR4
– 0xFFF27124–0xFFF27127	reserved	
– 0xFFF27128	Hardware Semaphore Register 5	HSMPR5
– 0xFFF2712C–0xFFF2712F	reserved	
– 0xFFF27130	Hardware Semaphore Register 6	HSMPR6

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF27134–0xFFF27137	reserved	
– 0xFFF27138	Hardware Semaphore Register 7	HSMPR7
– 0xFFF2713C–0xFFF271FF	reserved	
• 0xFFF27200–0xFFF272FF	GPIO (see **Chapter 22**, *GPIO*)	
– 0xFFF27200	Pin Open-Drain Register	PODR
– 0xFFF27204–0xFFF27207	reserved	
– 0xFFF27208	Pin Data Register	PDAT
– 0xFFF2720C–0xFFF2720F	reserved	
– 0xFFF27210	Pin Data Direction Register	PDIR
– 0xFFF27214–0xFFF27217	reserved	
– 0xFFF27218	Pin Assignment Register	PAR
– 0xFFF2721C–0xFFF2721F	reserved	
– 0xFFF27220	Pin Special Options Register	PSOR
– 0xFFF27224–0xFFF272FF	reserved	
• 0xFFF27300–0xFFF27FFF	reserved	
• 0xFFF28000–0xFFF281FF	General Configuration Registers (see **Chapter 8**, *General Configuration Registers*)	
– 0xFFF28000	General Configuration Register 1	GCR1
– 0xFFF28004	General Configuration Register 2	GCR2
– 0xFFF28008	General Status Register 1	GSR1
– 0xFFF2800C	High Speed Serial Interface Status Register	HSSI_SR
– 0xFFF28010	DDR General Configuration Register	DDR_GCR
– 0xFFF28014	High Speed Serial Interface Control Register 1	HSSI_CR1
– 0xFFF28018	High Speed Serial Interface Control Register 2	HSSI_CR2
– 0xFFF2801C	QUICC Engine Control Register	QECR
– 0xFFF28020	GPIO Pull-Up Enable Register	GPUER
– 0xFFF28024	GPIO Input Enable Register	GIER
– 0xFFF28028	System Part and Revision ID Register	SPRIDR
– 0xFFF2802C–0xFFF2802F	reserved	
– 0xFFF28030	General Control Register 4	GCR4
– 0xFFF28034	General Control Register 5	GCR5
– 0xFFF28038	General Status Register 2	GSR2
– 0xFFF2803C	Core Subsystem Slave Port Priority Control Register	TSPPCR
– 0xFFF28040	QUICC Engine First External Request Multiplex Register	CPCE1R
– 0xFFF28044	QUICC Engine Second External Request Multiplex Register	CPCE2R

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF28048	QUICC Engine Third External Request Multiplex Register	CPCE3R
– 0xFFF2804C	QUICC Engine Fourth External Request Multiplex Register	CPCE4R
– 0xFFF28050–0xFFF28073	reserved	
– 0xFFF28074	General Control Register 10	GCR10
– 0xFFF28078–0xFFF2807F	reserved	
– 0xFFF28080	General Interrupt Register 1	GIR1
– 0xFFF28084	General Interrupt Enable Register 1 for Core 0	GIER1_0
– 0xFFF28088	General Interrupt Enable Register 1 for Core 1	GIER1_1
– 0xFFF2808C	General Interrupt Enable Register 1 for Core 2	GIER1_2
– 0xFFF28090	General Interrupt Enable Register 1 for Core 3	GIER1_3
– 0xFFF28094	General Interrupt Enable Register 1 for Core 4	GIER1_4
– 0xFFF28098	General Interrupt Enable Register 1 for Core 5	GIER1_5
– 0xFFF2809C–0xFFF280A3	reserved	
– 0xFFF280A4	General Interrupt Register 3	GIR3
– 0xFFF280A8	General Interrupt Enable Register 3 for Core 0	GIER3_0
– 0xFFF280AC	General Interrupt Enable Register 3 for Core 1	GIER3_1
– 0xFFF280B0	General Interrupt Enable Register 3 for Core 2	GIER3_2
– 0xFFF280B4	General Interrupt Enable Register 3 for Core 3	GIER3_3
– 0xFFF280B8	General Interrupt Enable Register 3 for Core 4	GIER3_4
– 0xFFF280BC	General Interrupt Enable Register 3 for Core 5	GIER3_5
– 0xFFF280C0–0xFFF280EB	reserved	
– 0xFFF280EC	General Interrupt Register 5	GIR5
– 0xFFF280F0	General Interrupt Enable Register 5 for Core 0	GIER5_0
– 0xFFF280F4	General Interrupt Enable Register 5 for Core 1	GIER5_1
– 0xFFF280F8	General Interrupt Enable Register 5 for Core 2	GIER5_2
– 0xFFF280FC	General Interrupt Enable Register 5 for Core 3	GIER5_3
– 0xFFF28100	General Interrupt Enable Register 5 for Core 4	GIER5_4
– 0xFFF28104	General Interrupt Enable Register 5 for Core 5	GIER5_5
– 0xFFF28108–0xFFF2810F	reserved	
– 0xFFF28110	General Control Register 11	GCR11
– 0xFFF28114	General Control Register 12	GCR12
– 0xFFF28118–0xFFF2811F	reserved	
– 0xFFF28120	DMA Request0 Control Register	GCR_DREQ0
– 0xFFF28124	DMA Request1 Control Register	GCR_DREQ1
– 0xFFF28128	DMA Done Control Register	GCR_DDONE
– 0xFFF2812C	DDR1 General Configuration Register	DDR1_GCR
– 0xFFF28130	DDR2 General Configuration Register	DDR2_GCR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF28134–0xFFF28137	reserved	
– 0xFFF28138	Core Subsystem Slave Port General Configuration Register	CORE_SLV_GCR
– 0xFFF2813C–0xFFF281FF	reserved	
• 0xFFF28200–0xFFF2FFFF	reserved	
• 0xFFF30000–0xFFF33FFF	TDM0 (see **Chapter 19**, *TDM Interface*)	
– 0xFFF30000–0xFFF307FF	TDM0 Receive Local Memory	
– 0xFFF30800–0xFFF30FFF	reserved	
– 0xFFF31000–0xFFF313FC	TDM0 Receive Channel Parameters Register 0–255	TDM0RCPR[0–255]
– 0xFFF31400–0xFFF317FF	reserved	
– 0xFFF31800–0xFFF31FFF	TDM0 Transmit Local Memory	
– 0xFFF32000–0xFFF327FF	reserved	
– 0xFFF32800–0xFFF32BFC	TDM0 Transmit Channel Parameters Register 0–255	TDM0TCPR[0–255]
– 0xFFF32C00–0xFFF33EFF	reserved	
– 0xFFF33F00	TDM0 Parity Control Register	TDM0PCR
– 0xFFF33F04–0xFFF33F07	reserved	
– 0xFFF33F08	TDM0 Parity Error Register	TDM0PER
– 0xFFF33F0C–0xFFF33F0F	reserved	
– 0xFFF33F10	TDM0 Transmit Force Register	TDM0TFR
– 0xFFF33F14–0xFFF33F17	reserved	
– 0xFFF33F18	TDM0 Receive Force Register	TDM0RFR
– 0xFFF33F1C–0xFFF33F1F	reserved	
– 0xFFF33F20	TDM0 Transmit Status Register	TDM0TSR
– 0xFFF33F24–0xFFF33F27	reserved	
– 0xFFF33F28	TDM0 Receive Status Register	TDM0RSR
– 0xFFF33F2C–0xFFF33F2F	reserved	
– 0xFFF33F30	TDM0 Adaptation Status Register	TDM0ASR
– 0xFFF33F34–0xFFF33F37	reserved	
– 0xFFF33F38	TDM0 Transmit Event Register	TDM0TER

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF33F3C–0xFFF33F3F	reserved	
– 0xFFF33F40	TDM0 Receive Event Register	TDM0RER
– 0xFFF33F44–0xFFF33F47	reserved	
– 0xFFF33F48	TDM0 Transmit Number of Buffers	TDM0TNB
– 0xFFF33F4C–0xFFF33F4F	reserved	
– 0xFFF33F50	TDM0 Receive Number of Buffers	TDM0RNB
– 0xFFF33F54–0xFFF33F57	reserved	
– 0xFFF33F58	TDM0 Transmit Data Buffer Displacement Register	TDM0TDBDR
– 0xFFF33F5C–0xFFF33F5F	reserved	
– 0xFFF33F60	TDM0 Receive Data Buffer Displacement Register	TDM0RDBDR
– 0xFFF33F64–0xFFF33F67	reserved	
– 0xFFF33F68	TDM0 Adaptation Sync Distance Register	TDM0ASDR
– 0xFFF33F6C–0xFFF33F6F	reserved	
– 0xFFF33F70	TDM0 Transmit Interrupt Enable Register	TDM00TIER
– 0xFFF33F74–0xFFF33F77	reserved	
– 0xFFF33F78	TDM0 Receive Interrupt Enable Register	TDM0RIER
– 0xFFF33F7C–0xFFF33F7F	reserved	
– 0xFFF33F80	TDM0 Transmit Data Buffer Second Threshold	TDM0TDBST
– 0xFFF33F84–0xFFF33F87	reserved	
– 0xFFF33F88	TDM0 Receive Data Buffer Second Threshold	TDM0RDBST
– 0xFFF33F8C–0xFFF33F8F	reserved	
– 0xFFF33F90	TDM0 Transmit Data Buffer First Threshold	TDM0TDBFT
– 0xFFF33F94–0xFFF33F97	reserved	
– 0xFFF33F98	TDM0 Receive Data Buffer First Threshold	TDM0RDBFT
– 0xFFF33F9C–0xFFF33F9F	reserved	
– 0xFFF33FA0	TDM0 Transmit Control Register	TDM0TCR
– 0xFFF33FA4–0xFFF33FA7	reserved	
– 0xFFF33FA8	TDM0 Receive Control Register	TDM0RCR
– 0xFFF33FAC–0xFFF33FAF	reserved	
– 0xFFF33FB0	TDM0 Adaptation Control Register	TDM0ACR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF33FB4–0xFFF33FB7	reserved	
– 0xFFF33FB8	TDM0 Transmit Global Base Address	TDM0TGBA
– 0xFFF33FBC–0xFFF33FBF	reserved	
– 0xFFF33FC0	TDM0 Receive Global Base Address	TDM0RGBA
– 0xFFF33FC4–0xFFF33FC7	reserved	
– 0xFFF33FC8	TDM0 Transmit Data Buffer Size	TDM0TDBS
– 0xFFF33FCC–0xFFF33FCF	reserved	
– 0xFFF33FD0	TDM0 Receive Data Buffer Size	TDM0RDBS
– 0xFFF33FD4–0xFFF33FD7	reserved	
– 0xFFF33FD8	TDM0 Transmit Frame Parameters	TDM0TFP
– 0xFFF33FDC–0xFFF33FDF	reserved	
– 0xFFF33FE0	TDM0 Receive Frame Parameters	TDM0RFP
– 0xFFF33FE4–0xFFF33FE7	reserved	
– 0xFFF33FE8	TDM0 Transmit Interface Register	TDM0TIR
– 0xFFF33FEC–0xFFF33FEF	reserved	
– 0xFFF33FF0	TDM0 Receive Interface Register	TDM0RIR
– 0xFFF33FF4–0xFFF33FF7	reserved	
– 0xFFF33FF8	TDM0 General Interface Register	TDM0GIR
– 0xFFF33FFC–0xFFF33FFF	reserved	
• 0xFFF34000–0xFFF37FFF	TDM1 (see **Chapter 19**, *TDM Interface*)	
– 0xFFF34000–0xFFF347FF	TDM1 Receive Local Memory	
– 0xFFF34800–0xFFF34FFF	reserved	
– 0xFFF35000–0xFFF353FC	TDM1 Receive Channel Parameters Register 0–255	TDM1RCPR[0–255]
– 0xFFF35400–0xFFF357FF	reserved	
– 0xFFF35800–0xFFF35FFF	TDM1 Transmit Local Memory	
– 0xFFF36000–0xFFF367FF	reserved	
– 0xFFF36800–0xFFF36BFC	TDM1 Transmit Channel Parameters Register 0–255	TDM1TCPR[0–255]
– 0xFFF36C00–0xFFF37EFF	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF37F00	TDM1 Parity Control Register	TDM1PCR
– 0xFFF37F04–0xFFF37F07	reserved	
– 0xFFF37F08	TDM1 Parity Error Register	TDM1PER
– 0xFFF37F0C–0xFFF37F0F	reserved	
– 0xFFF37F10	TDM1 Transmit Force Register	TDM1TFR
– 0xFFF37F14–0xFFF37F17	reserved	
– 0xFFF37F18	TDM1 Receive Force Register	TDM1RFR
– 0xFFF37F1C–0xFFF37F1F	reserved	
– 0xFFF37F20	TDM1 Transmit Status Register	TDM1TSR
– 0xFFF37F24–0xFFF37F27	reserved	
– 0xFFF37F28	TDM1 Receive Status Register	TDM1RSR
– 0xFFF37F2C–0xFFF37F2F	reserved	
– 0xFFF37F30	TDM1 Adaptation Status Register	TDM1ASR
– 0xFFF37F34–0xFFF37F37	reserved	
– 0xFFF37F38	TDM1 Transmit Event Register	TDM1TER
– 0xFFF37F3C–0xFFF37F3F	reserved	
– 0xFFF37F40	TDM1 Receive Event Register	TDM1RER
– 0xFFF37F44–0xFFF37F47	reserved	
– 0xFFF37F48	TDM1 Transmit Number of Buffers	TDM1TNB
– 0xFFF37F4C–0xFFF37F4F	reserved	
– 0xFFF37F50	TDM1 Receive Number of Buffers	TDM1RNB
– 0xFFF37F54–0xFFF37F57	reserved	
– 0xFFF37F58	TDM1 Transmit Data Buffer Displacement Register	TDM1TDBDR
– 0xFFF37F5C–0xFFF37F5F	reserved	
– 0xFFF37F60	TDM1 Receive Data Buffer Displacement Register	TDM1RDBDR
– 0xFFF37F64–0xFFF37F67	reserved	
– 0xFFF37F68	TDM1 Adaptation Sync Distance Register	TDM1ASDR
– 0xFFF37F6C–0xFFF37F6F	reserved	
– 0xFFF37F70	TDM1 Transmit Interrupt Enable Register	TDM10TIER
– 0xFFF37F74–0xFFF37F77	reserved	
– 0xFFF37F78	TDM1 Receive Interrupt Enable Register	TDM1RIER

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF37F7C–0xFFF37F7F	reserved	
– 0xFFF37F80	TDM1 Transmit Data Buffer Second Threshold	TDM1TDBST
– 0xFFF37F84–0xFFF37F87	reserved	
– 0xFFF37F88	TDM1 Receive Data Buffer Second Threshold	TDM1RDBST
– 0xFFF37F8C–0xFFF37F8F	reserved	
– 0xFFF37F90	TDM1 Transmit Data Buffer First Threshold	TDM1TDBFT
– 0xFFF37F94–0xFFF37F97	reserved	
– 0xFFF37F98	TDM1 Receive Data Buffer First Threshold	TDM1RDBFT
– 0xFFF37F9C–0xFFF37F9F	reserved	
– 0xFFF37FA0	TDM1 Transmit Control Register	TDM1TCR
– 0xFFF37FA4–0xFFF37FA7	reserved	
– 0xFFF37FA8	TDM1 Receive Control Register	TDM1RCR
– 0xFFF37FAC–0xFFF37FAF	reserved	
– 0xFFF37FB0	TDM1 Adaptation Control Register	TDM1ACR
– 0xFFF37FB4–0xFFF37FB7	reserved	
– 0xFFF37FB8	TDM1 Transmit Global Base Address	TDM1TGBA
– 0xFFF37FBC–0xFFF37FBF	reserved	
– 0xFFF37FC0	TDM1 Receive Global Base Address	TDM1RGBA
– 0xFFF37FC4–0xFFF37FC7	reserved	
– 0xFFF37FC8	TDM1 Transmit Data Buffer Size	TDM1TDBS
– 0xFFF37FCC–0xFFF37FCF	reserved	
– 0xFFF37FD0	TDM1 Receive Data Buffer Size	TDM1RDBS
– 0xFFF37FD4–0xFFF37FD7	reserved	
– 0xFFF37FD8	TDM1 Transmit Frame Parameters	TDM1TFP
– 0xFFF37FDC–0xFFF37FDF	reserved	
– 0xFFF37FE0	TDM1 Receive Frame Parameters	TDM1RFP
– 0xFFF37FE4–0xFFF37FE7	reserved	
– 0xFFF37FE8	TDM1 Transmit Interface Register	TDM1TIR
– 0xFFF37FEC–0xFFF37FEF	reserved	
– 0xFFF37FF0	TDM1 Receive Interface Register	TDM1RIR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF37FF4–0xFFF37FF7	reserved	
– 0xFFF37FF8	TDM1 General Interface Register	TDM1GIR
– 0xFFF37FFC–0xFFF37FFF	reserved	
• 0xFFF38000–0xFFF3BFFF	TDM2 (see **Chapter 19**, *TDM Interface*)	
– 0xFFF38000–0xFFF387FF	TDM2 Receive Local Memory	
– 0xFFF38800–0xFFF38FFF	reserved	
– 0xFFF39000–0xFFF393FC	TDM2 Receive Channel Parameters Register 0–255	TDM2RCPR[0–255]
– 0xFFF39400–0xFFF397FF	reserved	
– 0xFFF39800–0xFFF39FFF	TDM2 Transmit Local Memory	
– 0xFFF3A000–0xFFF3A7FF	reserved	
– 0xFFF3A800–0xFFF3ABFC	TDM2 Transmit Channel Parameters Register 0–255	TDM2TCPR[0–255]
– 0xFFF3AC00–0xFFF3BEFF	reserved	
– 0xFFF3BF00	TDM2 Parity Control Register	TDM2PCR
– 0xFFF3BF04–0xFFF3BF07	reserved	
– 0xFFF3BF08	TDM2 Parity Error Register	TDM2PER
– 0xFFF3BF0C–0xFFF3BF0F	reserved	
– 0xFFF3BF10	TDM2 Transmit Force Register	TDM2TFR
– 0xFFF3BF14–0xFFF3BF17	reserved	
– 0xFFF3BF18	TDM2 Receive Force Register	TDM2RFR
– 0xFFF3BF1C–0xFFF3BF1F	reserved	
– 0xFFF3BF20	TDM2 Transmit Status Register	TDM2TSR
– 0xFFF3BF24–0xFFF3BF27	reserved	
– 0xFFF3BF28	TDM2 Receive Status Register	TDM2RSR
– 0xFFF3BF2C–0xFFF3BF2F	reserved	
– 0xFFF3BF30	TDM2 Adaptation Status Register	TDM2ASR
– 0xFFF3BF34–0xFFF38F37	reserved	
– 0xFFF3BF38	TDM2 Transmit Event Register	TDM2TER
– 0xFFF3BF3C–0xFFF3BF3F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF3BF40	TDM2 Receive Event Register	TDM2RER
– 0xFFF3BF44–0xFFF3BF47	reserved	
– 0xFFF3BF48	TDM2 Transmit Number of Buffers	TDM2TNB
– 0xFFF3BF4C–0xFFF3BF4F	reserved	
– 0xFFF3BF50	TDM2 Receive Number of Buffers	TDM2RNB
– 0xFFF3BF54–0xFFF3BF57	reserved	
– 0xFFF3BF58	TDM2 Transmit Data Buffer Displacement Register	TDM2TDBDR
– 0xFFF3BF5C–0xFFF3BF5F	reserved	
– 0xFFF3BF60	TDM2 Receive Data Buffer Displacement Register	TDM2RDBDR
– 0xFFF3BF64–0xFFF3BF67	reserved	
– 0xFFF3BF68	TDM2 Adaptation Sync Distance Register	TDM2ASDR
– 0xFFF3BF6C–0xFFF3BF6F	reserved	
– 0xFFF3BF70	TDM2 Transmit Interrupt Enable Register	TDM20TIER
– 0xFFF3BF74–0xFFF3BF77	reserved	
– 0xFFF3BF78	TDM2 Receive Interrupt Enable Register	TDM2RIER
– 0xFFF3BF7C–0xFFF3BF7F	reserved	
– 0xFFF3BF80	TDM2 Transmit Data Buffer Second Threshold	TDM2TDBST
– 0xFFF3BF84–0xFFF3BF87	reserved	
– 0xFFF3BF88	TDM2 Receive Data Buffer Second Threshold	TDM2RDBST
– 0xFFF3BF8C–0xFFF3BF8F	reserved	
– 0xFFF3BF90–0xFFF38F94	TDM2 Transmit Data Buffer First Threshold	TDM2TDBFT
– 0xFFF3BF98	TDM2 Receive Data Buffer First Threshold	TDM2RDBFT
– 0xFFF3BF9C–0xFFF3BF9F	reserved	
– 0xFFF3BFA0	TDM2 Transmit Control Register	TDM2TCR
– 0xFFF3BFA4–0xFFF3BFA7	reserved	
– 0xFFF3BFA8	TDM2 Receive Control Register	TDM2RCR
– 0xFFF3BFAC–0xFFF3BFAF	reserved	
– 0xFFF3BFB0	TDM2 Adaptation Control Register	TDM2ACR
– 0xFFF3BFB4–0xFFF3BFB7	reserved	
– 0xFFF3BFB8	TDM2 Transmit Global Base Address	TDM2TGBA

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF3BFBC–0xFFF3BFBF	reserved	
– 0xFFF3BFC0	TDM2 Receive Global Base Address	TDM2RGBA
– 0xFFF3BFC4–0xFFF3BFC7	reserved	
– 0xFFF3BFC8	TDM2 Transmit Data Buffer Size	TDM2TDBS
– 0xFFF3BFCC–0xFFF3BFCF	reserved	
– 0xFFF3BFD0	TDM2 Receive Data Buffer Size	TDM2RDBS
– 0xFFF3BFD4–0xFFF3BFD7	reserved	
– 0xFFF3BFD8	TDM2 Transmit Frame Parameters	TDM2TFP
– 0xFFF3BFDC–0xFFF3BFDF	reserved	
– 0xFFF3BFE0	TDM2 Receive Frame Parameters	TDM2RFP
– 0xFFF3BFE4–0xFFF3BFE7	reserved	
– 0xFFF3BFE8	TDM2 Transmit Interface Register	TDM2TIR
– 0xFFF3BFEC–0xFFF3BFEF	reserved	
– 0xFFF3BFF0	TDM2 Receive Interface Register	TDM2RIR
– 0xFFF3BFF4–0xFFF3BFF7	reserved	
– 0xFFF3BFF8	TDM2 General Interface Register	TDM2GIR
– 0xFFF3BFFC–0xFFF3BFFF	reserved	
• 0xFFF3C000–0xFFF3FFFF	TDM3 (see **Chapter 19**, *TDM Interface*)	
– 0xFFF3C000–0xFFF3C7FF	TDM3 Receive Local Memory	
– 0xFFF3C800–0xFFF3CFFF	reserved	
– 0xFFF3D000–0xFFF3D3FC	TDM3 Receive Channel Parameters Register 0–255	TDM3RCPR[0–255]
– 0xFFF3D400–0xFFF3D7FF	reserved	
– 0xFFF3D800–0xFFF3DFFF	TDM3 Transmit Local Memory	
– 0xFFF3E000–0xFFF3E7FF	reserved	
– 0xFFF3E800–0xFFF3EBFC	TDM3 Transmit Channel Parameters Register 0–255	TDM3TCPR[0–255]
– 0xFFF3EC00–0xFFF3FEFF	reserved	
– 0xFFF3FF00	TDM3 Parity Control Register	TDM3PCR
– 0xFFF3FFF04–0xFFF3FF07	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF3FF08	TDM3 Parity Error Register	TDM3PER
– 0xFFF3FF0C–0xFFF3FF0F	reserved	
– 0xFFF3FF10	TDM3 Transmit Force Register	TDM3TFR
– 0xFFF3FF14–0xFFF3FF17	reserved	
– 0xFFF3FF18	TDM3 Receive Force Register	TDM3RFR
– 0xFFF3FF1C–0xFFF3FF1F	reserved	
– 0xFFF3FF20	TDM3 Transmit Status Register	TDM3TSR
– 0xFFF3FF24–0xFFF3FF27	reserved	
– 0xFFF3FF28	TDM3 Receive Status Register	TDM3RSR
– 0xFFF3FF2C–0xFFF3FF2F	reserved	
– 0xFFF3FF30	TDM3 Adaptation Status Register	TDM3ASR
– 0xFFF3FF34–0xFFF3FF37	reserved	
– 0xFFF3FF38	TDM3 Transmit Event Register	TDM3TER
– 0xFFF3FF3C–0xFFF3FF3F	reserved	
– 0xFFF3FF40	TDM3 Receive Event Register	TDM3RER
– 0xFFF3FF44–0xFFF3FF47	reserved	
– 0xFFF3FF48	TDM3 Transmit Number of Buffers	TDM3TNB
– 0xFFF3FF4C–0xFFF3FF4F	reserved	
– 0xFFF3FF50	TDM3 Receive Number of Buffers	TDM3RNB
– 0xFFF3FF54–0xFFF3FF57	reserved	
– 0xFFF3FF58	TDM3 Transmit Data Buffer Displacement Register	TDM3TDBDR
– 0xFFF3FF5C–0xFFF3FF5F	reserved	
– 0xFFF3FF60	TDM3 Receive Data Buffer Displacement Register	TDM3RDBDR
– 0xFFF3FF64–0xFFF3FF67	reserved	
– 0xFFF3FF68	TDM3 Adaptation Sync Distance Register	TDM3ASDR
– 0xFFF3FF6C–0xFFF3FF6F	reserved	
– 0xFFF3FF70	TDM3 Transmit Interrupt Enable Register	TDM30TIER
– 0xFFF3FF74–0xFFF3FF77	reserved	
– 0xFFF3FF78	TDM3 Receive Interrupt Enable Register	TDM3RIER
– 0xFFF3FF7C–0xFFF3FF7F	reserved	
– 0xFFF3FF80	TDM3 Transmit Data Buffer Second Threshold	TDM3TDBST

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF3FF84–0xFFF3FF87	reserved	
– 0xFFF3FF88	TDM3 Receive Data Buffer Second Threshold	TDM3RDBST
– 0xFFF3FF8C–0xFFF3FF8F	reserved	
– 0xFFF3FF90	TDM3 Transmit Data Buffer First Threshold	TDM3TDBFT
– 0xFFF3FF94–0xFFF3FF97	reserved	
– 0xFFF3FF98	TDM3 Receive Data Buffer First Threshold	TDM3RDBFT
– 0xFFF3FF9C–0xFFF3FF9F	reserved	
– 0xFFF3FFA0	TDM3 Transmit Control Register	TDM3TCR
– 0xFFF3FFA4–0xFFF3FFA7	reserved	
– 0xFFF3FFA8	TDM3 Receive Control Register	TDM3RCR
– 0xFFF3FFAC–0xFFF3FFAF	reserved	
– 0xFFF3FFB0	TDM3 Adaptation Control Register	TDM3ACR
– 0xFFF3FFB4–0xFFF3FFB7	reserved	
– 0xFFF3FFB8	TDM3 Transmit Global Base Address	TDM3TGBA
– 0xFFF3FFBC–0xFFF3FFBF	reserved	
– 0xFFF3FFC0	TDM3 Receive Global Base Address	TDM3RGBA
– 0xFFF3FFC4–0xFFF3FFC7	reserved	
– 0xFFF3FFC8	TDM3 Transmit Data Buffer Size	TDM3TDBS
– 0xFFF3FFCC–0xFFF3FFCF	reserved	
– 0xFFF3FFD0	TDM3 Receive Data Buffer Size	TDM3RDBS
– 0xFFF3FFD4–0xFFF3FFD7	reserved	
– 0xFFF3FFD8	TDM3 Transmit Frame Parameters	TDM3TFP
– 0xFFF3FFDC–0xFFF3FFDF	reserved	
– 0xFFF3FFE0	TDM3 Receive Frame Parameters	TDM3RFP
– 0xFFF3FFE4–0xFFF3FFE7	reserved	
– 0xFFF3FFE8	TDM3 Transmit Interface Register	TDM3TIR
– 0xFFF3FFEC–0xFFF3FFEF	reserved	
– 0xFFF3FFF0	TDM3 Receive Interface Register	TDM3RIR
– 0xFFF3FFF4–0xFFF3FFF7	reserved	
– 0xFFF3FFF8	TDM3 General Interface Register	TDM3GIR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF3FFFC–0xFFF3FFFF	reserved	
• 0xFFF40000–0xFFF7FFFF	reserved	
• 0xFFF80000–0xFFF9FFFF	RapidIO (see **Chapter 16**, *Serial RapidIO Controller*)	
– 0xFFF80000	Device Identity Capability Register	DIDCAR
– 0xFFF80004	Device Information Capability Register	DICAR
– 0xFFF80008	Assembly Identity Capability Register	AIDCAR
– 0xFFF8000C	Assembly Information Capability Register	AICAR
– 0xFFF80010	Processing Element Features Capability Register	PEFCAR
– 0xFFF80018	Source Operations Capability Register	SOCAR
– 0xFFF8001C	Destination Operations Capability Register	DOCAR
– 0xFFF80020–0xFFF8003F	reserved	
– 0xFFF80040	Mailbox Command And Status Register	MCSR
– 0xFFF80044	Port -Write and Doorbell Command and Status Register	PWDCSR
– 0xFFF80048–0xFFF8004B	reserved	
– 0xFFF8004C	Processing Element Logical Layer Control Command and Status Register	PELLCCSR
– 0xFFF80050–0xFFF8005B	reserved	
– 0xFFF8005C	Local Configuration Space Base Address 1 Command and Status Register	LCSBA1CSR
– 0xFFF80060	Base Device ID Command and Status Register	BDIDCSR
– 0xFFF80064–0xFFF80067	reserved	
– 0xFFF80068	Host Base Device ID Lock Command and Status Register	HBDIDLCSR
– 0xFFF8006C	Component Tag Command and Status Register	CTCSR
– 0xFFF80070–0xFFF800FF	reserved	
– 0xFFF80100	Port Maintenance Block Header 0	PMBH0
– 0xFFF80104–0xFFF8011F	reserved	
– 0xFFF80120	Port Link Time-out Control Command and Status Register	PLTOCCSR
– 0xFFF80124	Port Response Time-out Control Command and Status Register	PRTOCCSR
– 0xFFF80128–0xFFF8013B	reserved	
– 0xFFF8013C	Port General Control Command and Status Register	PGCCSR
– 0xFFF80140	Port 0 Link Maintenance Request Command and Status Register	P0LMREQCSR
– 0xFFF80144	Port 0 Link Maintenance Response Command and Status Register	P0LMRESPCSR
– 0xFFF80148	Port 0 Local ackID Status Command and Status Register	P0LASCSR
– 0xFFF8014C–0xFFF80157	reserved	
– 0xFFF80158	Port 0 Error and Status Command and Status Register	P0ESCSR
– 0xFFF8015C	Port 0 Control Command and Status Register	P0CCSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF80160	Port 1 Link Maintenance Request Command and Status Register	P1LMREQCSR
– 0xFFF80164	Port 1 Link Maintenance Response Command and Status Register	P1LMRESPCSR
– 0xFFF80168	Port 1 Local ackID Status Command and Status Register	P1LASCSR
– 0xFFF8016C–0xFFF80177	reserved	
– 0xFFF80178	Port 1 Error and Status Command and Status Register	P1ESCSR
– 0xFFF8017C	Port 1 Control Command and Status Register	P1CCSR
– 0xFFF80180–0xFFF805FF	reserved	
– 0xFFF80600	Error Reporting Block Header	ERBH
– 0xFFF80604–0xFFF80607	reserved	
– 0xFFF80608	Logical/Transport Layer Error Detect Command and Status Register	LTLEDCSR
– 0xFFF8060C	Logical/Transport Layer Error Enable Command and Status Register	LTLEECSR
– 0xFFF80610–0xFFF80613	reserved	
– 0xFFF80614	Logical/Transport Layer Address Capture Command and Status Register	LTLACCSR
– 0xFFF80618	Logical/Transport Layer Device ID Capture Command and Status Register	LTLDIDCCSR
– 0xFFF8061C	Logical/Transport Layer Control Capture Command and Status Register	LTLCCCSR
– 0xFFF80620–0xFFF8063F	reserved	
– 0xFFF80640	Port 0 Error Detect Command and Status Register	P0EDCSR
– 0xFFF80644	Port 0 Error Rate Enable Command and Status Register	P0ERECSR
– 0xFFF80648	Port 0 Error Capture Attributes Command and Status Register	P0ECACSR
– 0xFFF8064C	Port 0 Packet/Control Symbol Error Capture Command and Status Register 0	P0PCSECCSR0
– 0xFFF80650	Port 0 Packet Error Capture Command and Status Register 1	P0PECCSR1
– 0xFFF80654	Port 0 Packet Error Capture Command and Status Register 2	P0PECCSR2
– 0xFFF80658	Port 0 Packet Error Capture Command and Status Register 3	P0PECCSR3
– 0xFFF8065C–0xFFF80667	reserved	
– 0xFFF80668	Port 0 Error Rate Command and Status Register	P0ERCSR
– 0xFFF8066C	Port 0 Error Rate Threshold Command and Status Register	P0ERTCSR
– 0xFFF80670–0xFFF8067F	reserved	
– 0xFFF80680	Port 1 Error Detect Command and Status Register	P1EDCSR
– 0xFFF80684	Port 1 Error Rate Enable Command and Status Register	P1ERECSR
– 0xFFF80688	Port 1 Error Capture Attributes Command and Status Register	P1ECACSR
– 0xFFF8068C	Port 1 Packet/Control Symbol Error Capture Command and Status Register 0	P1PCSECCSR0
– 0xFFF80690	Port 1 Packet Error Capture Command and Status Register 1	P1PECCSR1
– 0xFFF80694	Port 1 Packet Error Capture Command and Status Register 2	P1PECCSR2
– 0xFFF80698	Port 1 Packet Error Capture Command and Status Register 3	P1PECCSR3
– 0xFFF8069C–0xFFF806A7	reserved	
– 0xFFF806A8	Port 1 Error Rate Command and Status Register	P1ERCSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF806AC	Port 1 Error Rate Threshold Command and Status Register	P1ERTCSR
– 0xFFF806B0–0xFFF90003	reserved	
– 0xFFF90004	Logical Layer Configuration Register	LLCR
– 0xFFF90008–0xFFF9000F	reserved	
– 0xFFF90010	Error/Port-Write Interrupt Status Register	EPWISR
– 0xFFF90014–0xFFF9001F	reserved	
– 0xFFF90020	Logical Retry Error Threshold Configuration Register	LRETCR
– 0xFFF90024–0xFFF9007F	reserved	
– 0xFFF90080	Physical Retry Error Threshold Configuration Register	PRETCR
– 0xFFF90084–0xFFF900FF	reserved	
– 0xFFF90100	Port 0 Alternate Device ID Command and Status Register	P0ADIDCSR
– 0xFFF90104–0xFFF9011F	reserved	
– 0xFFF90120	Port 0 Pass-Through Accept-All Configuration Register	P0PTAACR
– 0xFFF90124	Port 0 Logical Outbound Packet Time-to-Live Configuration Register	P0LOPTTLCR
– 0xFFF90128–0xFFF9012F	reserved	
– 0xFFF90130	Port 0 Implementation Error Command and Status Register	P0IECSR
– 0xFFF90134–0xFFF9013F	reserved	
– 0xFFF90140	Port 0 Physical Configuration Register	P0PCR
– 0xFFF90144–0xFFF90157	reserved	
– 0xFFF90158	Port 0 Serial Link Command And Status Register	P0SLCSR
– 0xFFF9015C–0xFFF9015F	reserved	
– 0xFFF90160	Port 0 Serial Link Error Injection Configuration Register	P0SLEICR
– 0xFFF90164–0xFFF9017F	reserved	
– 0xFFF90180	Port 1 Alternate Device ID Command and Status Register	P1ADIDCSR
– 0xFFF90184–0xFFF9019F	reserved	
– 0xFFF901A0	Port 1 Pass-Through Accept-All Configuration Register	P1PTAACR
– 0xFFF901A4	Port 1 Logical Outbound Packet Time-to-Live Configuration Register	P1LOPTTLCR
– 0xFFF901A8–0xFFF901AF	reserved	
– 0xFFF901B0	Port 1 Implementation Error Command and Status Register	P1IECSR
– 0xFFF901B4–0xFFF901BF	reserved	
– 0xFFF901C0	Port 1 Physical Configuration Register	P1PCR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF901C4–0xFFF901D7	reserved	
– 0xFFF901D8	Port 1 Serial Link Command And Status Register	P1SLCSR
– 0xFFF901DC–0xFFF901DF	reserved	
– 0xFFF901E0	Port 1 Serial Link Error Injection Configuration Register	P1SLEICR
– 0xFFF901E4–0xFFF90BF7	reserved	
– 0xFFF90BF8	IP Block Revision Register 1	IPBRR1
– 0xFFF90BFC	IP Block Revision Register 2	IPBRR2
– 0xFFF90C00	Port 0 RapidIO Outbound Window Translation Address Register 0	P0ROWTAR0
– 0xFFF90C04	Port 0 RapidIO Outbound Window Translation Extended Address Register 0	P0ROWTEAR0
– 0xFFF90C08–0xFFF90C0F	reserved	
– 0xFFF90C10	Port 0 RapidIO Outbound Window Attributes Register 0	P0ROWAR0
– 0xFFF90C14–0xFFF90C1F	reserved	
– 0xFFF90C20	Port 0 RapidIO Outbound Window Translation Address Register 1	P0ROWTAR1
– 0xFFF90C24	Port 0 RapidIO Outbound Window Translation Extended Address Register 1	P0ROWTEAR1
– 0xFFF90C28–0xFFF90C2F	reserved	
– 0xFFF90C30	Port 0 RapidIO Outbound Window Attributes Register 1	P0ROWAR1
– 0xFFF90C34	Port 0 RapidIO Outbound Window Segment 1 Register 1	P0ROWS1R1
– 0xFFF90C38	Port 0 RapidIO Outbound Window Segment 2 Register 1	P0ROWS2R1
– 0xFFF90C3C	Port 0 RapidIO Outbound Window Segment 3 Register 1	P0ROWS3R1
– 0xFFF90C40	Port 0 RapidIO Outbound Window Translation Address Register 2	P0ROWTAR2
– 0xFFF90C44	Port 0 RapidIO Outbound Window Translation Extended Address Register 2	P0ROWTEAR2
– 0xFFF90C48	Port 0 RapidIO Outbound Window Base Address Register 2	P0ROWBAR2
– 0xFFF90C4C–0xFFF90C4F	reserved	
– 0xFFF90C50	Port 0 RapidIO Outbound Window Attributes Register 2	P0ROWAR2
– 0xFFF90C54	Port 0 RapidIO Outbound Window Segment 1 Register 2	P0ROWS1R2
– 0xFFF90C58	Port 0 RapidIO Outbound Window Segment 2 Register 2	P0ROWS2R2
– 0xFFF90C5C	Port 0 RapidIO Outbound Window Segment 3 Register 2	P0ROWS3R2
– 0xFFF90C60	Port 0 RapidIO Outbound Window Translation Address Register 3	P0ROWTAR3
– 0xFFF90C64	Port 0 RapidIO Outbound Window Translation Extended Address Register 3	P0ROWTEAR3
– 0xFFF90C68	Port 0 RapidIO Outbound Window Base Address Register 3	P0ROWBAR3
– 0xFFF90C6C–0xFFF90C6F	reserved	
– 0xFFF90C70	Port 0 RapidIO Outbound Window Attributes Register 3	P0ROWAR3
– 0xFFF90C74	Port 0 RapidIO Outbound Window Segment 1 Register 3	P0ROWS1R3
– 0xFFF90C78	Port 0 RapidIO Outbound Window Segment 2 Register 3	P0ROWS2R3
– 0xFFF90C7C	Port 0 RapidIO Outbound Window Segment 3 Register 3	P0ROWS3R3
– 0xFFF90C80	Port 0 RapidIO Outbound Window Translation Address Register 4	P0ROWTAR4

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90C84	Port 0 RapidIO Outbound Window Translation Extended Address Register 4	P0ROWTEAR4
– 0xFFF90C88	Port 0 RapidIO Outbound Window Base Address Register 4	P0ROWBAR4
– 0xFFF90C8C–0xFFF90C8F	reserved	
– 0xFFF90C90	Port 0 RapidIO Outbound Window Attributes Register 4	P0ROWAR4
– 0xFFF90C94	Port 0 RapidIO Outbound Window Segment 1 Register 4	P0ROWS1R4
– 0xFFF90C98	Port 0 RapidIO Outbound Window Segment 2 Register 4	P0ROWS2R4
– 0xFFF90C9C	Port 0 RapidIO Outbound Window Segment 3 Register 4	P0ROWS3R4
– 0xFFF90CA0	Port 0 RapidIO Outbound Window Translation Address Register 5	P0ROWTAR5
– 0xFFF90CA4	Port 0 RapidIO Outbound Window Translation Extended Address Register 5	P0ROWTEAR5
– 0xFFF90CA8	Port 0 RapidIO Outbound Window Base Address Register 5	P0ROWBAR5
– 0xFFF90CAC–0xFFF90CAF	reserved	
– 0xFFF90CB0	Port 0 RapidIO Outbound Window Attributes Register 5	P0ROWAR5
– 0xFFF90CB4	Port 0 RapidIO Outbound Window Segment 1 Register 5	P0ROWS1R5
– 0xFFF90CB8	Port 0 RapidIO Outbound Window Segment 2 Register 5	P0ROWS2R5
– 0xFFF90CBC	Port 0 RapidIO Outbound Window Segment 3 Register 5	P0ROWS3R5
– 0xFFF90CC0	Port 0 RapidIO Outbound Window Translation Address Register 6	P0ROWTAR6
– 0xFFF90CC4	Port 0 RapidIO Outbound Window Translation Extended Address Register 6	P0ROWTEAR6
– 0xFFF90CC8	Port 0 RapidIO Outbound Window Base Address Register 6	P0ROWBAR6
– 0xFFF90CCC–0xFFF90CCF	reserved	
– 0xFFF90CD0	Port 0 RapidIO Outbound Window Attributes Register 6	P0ROWAR6
– 0xFFF90CD4	Port 0 RapidIO Outbound Window Segment 1 Register 6	P0ROWS1R6
– 0xFFF90CD8	Port 0 RapidIO Outbound Window Segment 2 Register 6	P0ROWS2R6
– 0xFFF90CDC	Port 0 RapidIO Outbound Window Segment 3 Register 6	P0ROWS3R6
– 0xFFF90CE0	Port 0 RapidIO Outbound Window Translation Address Register 7	P0ROWTAR7
– 0xFFF90CE4	Port 0 RapidIO Outbound Window Translation Extended Address Register 7	P0ROWTEAR7
– 0xFFF90CE8	Port 0 RapidIO Outbound Window Base Address Register 7	P0ROWBAR7
– 0xFFF90CEC–0xFFF90CEF	reserved	
– 0xFFF90CF0	Port 0 RapidIO Outbound Window Attributes Register 7	P0ROWAR7
– 0xFFF90CF4	Port 0 RapidIO Outbound Window Segment 1 Register 7	P0ROWS1R7
– 0xFFF90CF8	Port 0 RapidIO Outbound Window Segment 2 Register 7	P0ROWS2R7
– 0xFFF90CFC	Port 0 RapidIO Outbound Window Segment 3 Register 7	P0ROWS3R7
– 0xFFF90D00	Port 0 RapidIO Outbound Window Translation Address Register 8	P0ROWTAR8
– 0xFFF90D04	Port 0 RapidIO Outbound Window Translation Extended Address Register 8	P0ROWTEAR8
– 0xFFF90D08	Port 0 RapidIO Outbound Window Base Address Register 8	P0ROWBAR8
– 0xFFF90D0C–0xFFF90D0F	reserved	
– 0xFFF90D10	Port 0 RapidIO Outbound Window Attributes Register 8	P0ROWAR8
– 0xFFF90D14	Port 0 RapidIO Outbound Window Segment 1 Register 8	P0ROWS1R8
– 0xFFF90D18	Port 0 RapidIO Outbound Window Segment 2 Register 8	P0ROWS2R8

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90D1C	Port 0 RapidIO Outbound Window Segment 3 Register 8	P0ROWS3R8
– 0xFFF90D20–0xFFF90D5F	reserved	
– 0xFFF90D60	Port 0 RapidIO Inbound Window Translation Address Register 4	P0RIWTAR4
– 0xFFF90D64–0xFFF90D67	reserved	
– 0xFFF90D68	Port 0 RapidIO Inbound Window Base Address Register 4	P0RIWBAR4
– 0xFFF90D6C–0xFFF90D6F	reserved	
– 0xFFF90D70	Port 0 RapidIO Inbound Window Attributes Register 4	P0RIWAR4
– 0xFFF90D74–0xFFF90D7F	reserved	
– 0xFFF90D80	Port 0 RapidIO Inbound Window Translation Address Register 3	P0RIWTAR3
– 0xFFF90D84–0xFFF90D87	reserved	
– 0xFFF90D88	Port 0 RapidIO Inbound Window Base Address Register 3	P0RIWBAR3
– 0xFFF90D8C–0xFFF90D8F	reserved	
– 0xFFF90D90	Port 0 RapidIO Inbound Window Attributes Register 3	P0RIWAR3
– 0xFFF90D94–0xFFF90D9F	reserved	
– 0xFFF90DA0	Port 0 RapidIO Inbound Window Translation Address Register 2	P0RIWTAR2
– 0xFFF90DA4–0xFFF90DA7	reserved	
– 0xFFF90DA8	Port 0 RapidIO Inbound Window Base Address Register 2	P0RIWBAR2
– 0xFFF90DAC–0xFFF90DAF	reserved	
– 0xFFF90DB0	Port 0 RapidIO inbound window attributes register 2	P0RIWAR2
– 0xFFF90DB4–0xFFF90DBF	reserved	
– 0xFFF90DC0	Port 0 RapidIO Inbound Window Translation Address Register 1	P0RIWTAR1
– 0xFFF90DC4–0xFFF90DC7	reserved	
– 0xFFF90DC8	Port 0 RapidIO Inbound Window Base Address Register 1	P0RIWBAR1
– 0xFFF90DCC–0xFFF90DCF	reserved	
– 0xFFF90DD0	Port 0 RapidIO Inbound Window Attributes Register 1	P0RIWAR1
– 0xFFF90DD4–0xFFF90DDF	reserved	
– 0xFFF90DE0	Port 0 RapidIO Inbound Window Translation Address Register 0	P0RIWTAR0
– 0xFFF90DE4–0xFFF90DEF	reserved	
– 0xFFF90DF0	Port 0 RapidIO Inbound Window Attributes Register 0	P0RIWAR0
– 0xFFF90DF4–0xFFF92DFF	reserved	
– 0xFFF90E00	Port 1 RapidIO Outbound Window Translation Address Register 0	P1ROWTAR0

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90E04	Port 1 RapidIO Outbound Window Translation Extended Address Register 0	P1ROWTEAR0
– 0xFFF90E08	Port 1 RapidIO Outbound Window Base Address Register 0	P1ROWBAR0
– 0xFFF90E0C–0xFFF90E0F	reserved	
– 0xFFF90E10	Port 1 RapidIO Outbound Window Attributes Register 0	P1ROWAR0
– 0xFFF90E14–0xFFF90E1F	reserved	
– 0xFFF90E20	Port 1 RapidIO Outbound Window Translation Address Register 1	P1ROWTAR1
– 0xFFF90E24	Port 1 RapidIO Outbound Window Translation Extended Address Register 1	P1ROWTEAR1
– 0xFFF90E28	Port 1 RapidIO Outbound Window Base Address Register 1	P1ROWBAR1
– 0xFFF90E2C–0xFFF90E2F	reserved	
– 0xFFF90E30	Port 1 RapidIO Outbound Window Attributes Register 1	P1ROWAR1
– 0xFFF90E34	Port 1 RapidIO Outbound Window Segment 1 Register 1	P1ROWS1R1
– 0xFFF90E38	Port 1 RapidIO Outbound Window Segment 2 Register 1	P1ROWS2R1
– 0xFFF90E3C	Port 1 RapidIO Outbound Window Segment 3 Register 1	P1ROWS3R1
– 0xFFF90E40	Port 1 RapidIO Outbound Window Translation Address Register 2	P1ROWTAR2
– 0xFFF90E44	Port 1 RapidIO Outbound Window Translation Extended Address Register 2	P1ROWTEAR2
– 0xFFF90E48	Port 1 RapidIO Outbound Window Base Address Register 2	P1ROWBAR2
– 0xFFF90E4C–0xFFF90E4F	reserved	
– 0xFFF90E50	Port 1 RapidIO Outbound Window Attributes Register 2	P1ROWAR2
– 0xFFF90E54	Port 1 RapidIO Outbound Window Segment 1 Register 2	P1ROWS1R2
– 0xFFF90E58	Port 1 RapidIO Outbound Window Segment 2 Register 2	P1ROWS2R2
– 0xFFF90E5C	Port 1 RapidIO Outbound Window Segment 3 Register 2	P1ROWS3R2
– 0xFFF90E60	Port 1 RapidIO Outbound Window Translation Address Register 3	P1ROWTAR3
– 0xFFF90E64	Port 1 RapidIO Outbound Window Translation Extended Address Register 3	P1ROWTEAR3
– 0xFFF90E68	Port 1 RapidIO Outbound Window Base Address Register 3	P1ROWBAR3
– 0xFFF90E6C–0xFFF90E6F	reserved	
– 0xFFF90E70	Port 1 RapidIO Outbound Window Attributes Register 3	P1ROWAR3
– 0xFFF90E74	Port 1 RapidIO Outbound Window Segment 1 Register 3	P1ROWS1R3
– 0xFFF90E78	Port 1 RapidIO Outbound Window Segment 2 Register 3	P1ROWS2R3
– 0xFFF90E7C	Port 1 RapidIO Outbound Window Segment 3 Register 3	P1ROWS3R3
– 0xFFF90E80	Port 1 RapidIO Outbound Window Translation Address Register 4	P1ROWTAR4
– 0xFFF90E84	Port 1 RapidIO Outbound Window Translation Extended Address Register 4	P1ROWTEAR4
– 0xFFF90E88	Port 1 RapidIO Outbound Window Base Address Register 4	P1ROWBAR4
– 0xFFF90E8C–0xFFF90E8F	reserved	
– 0xFFF90E90	Port 1 RapidIO Outbound Window Attributes Register 4	P1ROWAR4
– 0xFFF90E94	Port 1 RapidIO Outbound Window Segment 1 Register 4	P1ROWS1R4
– 0xFFF90E98	Port 1 RapidIO Outbound Window Segment 2 Register 4	P1ROWS2R4
– 0xFFF90E9C	Port 1 RapidIO Outbound Window Segment 3 Register 4	P1ROWS3R4

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90EA0	Port 1 RapidIO Outbound Window Translation Address Register 5	P1ROWTAR5
– 0xFFF90EA4	Port 1 RapidIO Outbound Window Translation Extended Address Register 5	P1ROWTEAR5
– 0xFFF90EA8	Port 1 RapidIO Outbound Window Base Address Register 5	P1ROWBAR5
– 0xFFF90EAC–0xFFF90EAF	reserved	
– 0xFFF90EB0	Port 1 RapidIO Outbound Window Attributes Register 5	P1ROWAR5
– 0xFFF90EB4	Port 1 RapidIO Outbound Window Segment 1 Register 5	P1ROWS1R5
– 0xFFF90EB8	Port 1 RapidIO Outbound Window Segment 2 Register 5	P1ROWS2R5
– 0xFFF90EBC	Port 1 RapidIO Outbound Window Segment 3 Register 5	P1ROWS3R5
– 0xFFF90EC0	Port 1 RapidIO Outbound Window Translation Address Register 6	P1ROWTAR6
– 0xFFF90EC4	Port 1 RapidIO Outbound Window Translation Extended Address Register 6	P1ROWTEAR6
– 0xFFF90EC8	Port 1 RapidIO Outbound Window Base Address Register 6	P1ROWBAR6
– 0xFFF90ECC–0xFFF90ECF	reserved	
– 0xFFF90ED0	Port 1 RapidIO Outbound Window Attributes Register 6	P1ROWAR6
– 0xFFF90ED4	Port 1 RapidIO Outbound Window Segment 1 Register 6	P1ROWS1R6
– 0xFFF90ED8	Port 1 RapidIO Outbound Window Segment 2 Register 6	P1ROWS2R6
– 0xFFF90EDC	Port 1 RapidIO Outbound Window Segment 3 Register 6	P1ROWS3R6
– 0xFFF90EE0	Port 1 RapidIO Outbound Window Translation Address Register 7	P1ROWTAR7
– 0xFFF90EE4	Port 1 RapidIO Outbound Window Translation Extended Address Register 7	P1ROWTEAR7
– 0xFFF90EE8	Port 1 RapidIO Outbound Window Base Address Register 7	P1ROWBAR7
– 0xFFF90EEC–0xFFF90EEF	reserved	
– 0xFFF90EF0	Port 1 RapidIO Outbound Window Attributes Register 7	P1ROWAR7
– 0xFFF90EF4	Port 1 RapidIO Outbound Window Segment 1 Register 7	P1ROWS1R7
– 0xFFF90EF8	Port 1 RapidIO Outbound Window Segment 2 Register 7	P1ROWS2R7
– 0xFFF90EFC	Port 1 RapidIO Outbound Window Segment 3 Register 7	P1ROWS3R7
– 0xFFF90F00	Port 1 RapidIO Outbound Window Translation Address Register 8	P1ROWTAR8
– 0xFFF90F04	Port 1 RapidIO Outbound Window Translation Extended Address Register 8	P1ROWTEAR8
– 0xFFF90F08	Port 1 RapidIO Outbound Window Base Address Register 8	P1ROWBAR8
– 0xFFF90F0C–0xFFF90F0F	reserved	
– 0xFFF90F10	Port 1 RapidIO Outbound Window Attributes Register 8	P1ROWAR8
– 0xFFF90F14	Port 1 RapidIO Outbound Window Segment 1 Register 8	P1ROWS1R8
– 0xFFF90F18	Port 1 RapidIO Outbound Window Segment 2 Register 8	P1ROWS2R8
– 0xFFF90F1C	Port 1 RapidIO Outbound Window Segment 3 Register 8	P1ROWS3R8
– 0xFFF90F20–0xFFF90F5F	reserved	
– 0xFFF90F60	Port 1 RapidIO Inbound Window Translation Address Register 4	P1RIWTAR4
– 0xFFF90F64–0xFFF90F67	reserved	
– 0xFFF90F68	Port 1 RapidIO Inbound Window Base Address Register 4	P1RIWBAR4

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF90F6C–0xFFF90F6F	reserved	
– 0xFFF90F70	Port 1 RapidIO Inbound Window Attributes Register 4	P1RIWAR4
– 0xFFF90F74–0xFFF90F7F	reserved	
– 0xFFF90F80	Port 1 RapidIO Inbound Window Translation Address Register 3	P1RIWTAR3
– 0xFFF90F84–0xFFF90F87	reserved	
– 0xFFF90F88	Port 1 RapidIO Inbound Window Base Address Register 3	P1RIWBAR3
– 0xFFF90F8C–0xFFF90F8F	reserved	
– 0xFFF90F90	Port 1 RapidIO Inbound Window Attributes Register 3	P1RIWAR3
– 0xFFF90F94–0xFFF90F9F	reserved	
– 0xFFF90FA0	Port 1 RapidIO Inbound Window Translation Address Register 2	P1RIWTAR2
– 0xFFF90FA4–0xFFF90FA7	reserved	
– 0xFFF90FA8	Port 1 RapidIO Inbound Window Base Address Register 2	P1RIWBAR2
– 0xFFF90FAC–0xFFF90FAF	reserved	
– 0xFFF90FB0	Port 1 RapidIO inbound window attributes register 2	P1RIWAR2
– 0xFFF90FB4–0xFFF90FBF	reserved	
– 0xFFF90FC0	Port 1 RapidIO Inbound Window Translation Address Register 1	P1RIWTAR1
– 0xFFF90FC4–0xFFF90FC7	reserved	
– 0xFFF90FC8	Port 1 RapidIO Inbound Window Base Address Register 1	P1RIWBAR1
– 0xFFF90FCC–0xFFF90FCF	reserved	
– 0xFFF90FD0	Port 1 RapidIO Inbound Window Attributes Register 1	P1RIWAR1
– 0xFFF90FD4–0xFFF90FDF	reserved	
– 0xFFF90FE0	Port 1 RapidIO Inbound Window Translation Address Register 0	P1RIWTAR0
– 0xFFF90FE4–0xFFF90FE7	reserved	
– 0xFFF90FE8	Port 1 RapidIO Inbound Window Base Address Register 1	P1RIWBAR1
– 0xFFF90FEC–0xFFF90FEF	reserved	
– 0xFFF90FF0	Port 1 RapidIO Inbound Window Attributes Register 0	P1RIWAR0
– 0xFFF90FF4–0xFFF92DFF	reserved	
– 0xFFF90FF4–0xFFF92FFF	reserved	
– 0xFFF93000	Outbound Message 0 Mode Register	OM0MR
– 0xFFF93004	Outbound Message 0 Status Register	OM0SR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF93008–0xFFF9300B	reserved	
– 0xFFF9300C	Outbound Message 0 Descriptor Queue Dequeue Pointer Address Register	OM0DQDPAR
– 0xFFF93010–0xFFF93013	reserved	
– 0xFFF93014	Outbound Message 0 Source Address Register	OM0SAR
– 0xFFF93018	Outbound Message 0 Destination Port Register	OM0DPR
– 0xFFF9301C	Outbound Message 0 Destination Attributes Register	OM0DATR
– 0xFFF93020	Outbound Message 0 Double-word Count Register	OM0DCR
– 0xFFF93024–0xFFF93027	reserved	
– 0xFFF93028	Outbound Message 0 Descriptor Queue Enqueue Pointer Address Register	OM0DQEPAR
– 0xFFF9302C	Outbound Message 0 Retry Error Threshold Configuration Register	OM0RETCR
– 0xFFF93030	Outbound Message 0 Multicast Group Register	OM0MGR
– 0xFFF93034	Outbound Message 0 Multicast List Register	OM0MLR
– 0xFFF93038–0xFFF9305F	reserved	
– 0xFFF93060	Inbound Message 0 Mode Register	IM0MR
– 0xFFF93064	Inbound Message 0 Status Register	IM0SR
– 0xFFF93068–0xFFF9306B	reserved	
– 0xFFF9306C	Inbound Message 0 Frame Queue Dequeue Pointer Address Register	IM0FQDPAR
– 0xFFF93070–0xFFF93073	reserved	
– 0xFFF93074	Inbound Message 0 Frame Queue Enqueue Pointer Address Register	IM0FQEPAR
– 0xFFF93078	Inbound Message 0 Maximum Interrupt Report Interval Register	IM0MIRIR
– 0xFFF9307C–0xFFF930FF	reserved	
– 0xFFF93100	Outbound Message 1 Mode Register	OM1MR
– 0xFFF93104	Outbound Message 1 Status Register	OM1SR
– 0xFFF93108–0xFFF9310B	reserved	
– 0xFFF9310C	Outbound Message 1 Descriptor Queue Dequeue Pointer Address Register	OM1DQDPAR
– 0xFFF93110–0xFFF93113	reserved	
– 0xFFF93114	Outbound Message 1 Source Address Register	OM1SAR
– 0xFFF93118	Outbound Message 1 Destination Port Register	OM1DPR
– 0xFFF9311C	Outbound Message 1 Destination Attributes Register	OM1DATR
– 0xFFF93120	Outbound Message 1 Double-word Count Register	OM1DCR
– 0xFFF93124–0xFFF93127	reserved	
– 0xFFF93128	Outbound Message 1 Descriptor Queue Enqueue Pointer Address Register	OM1DQEPAR
– 0xFFF9312C	Outbound Message 1 Retry Error Threshold Configuration Register	OM1RETCR
– 0xFFF93130	Outbound Message 1 Multicast Group Register	OM1MGR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFF93134	Outbound Message 1 Multicast List Register	OM1MLR
– 0xFFF93138–0xFFF9315F	reserved	
– 0xFFF93160	Inbound Message 1 Mode Register	IM1MR
– 0xFFF93164	Inbound Message 1 Status Register	IM1SR
– 0xFFF93168–0xFFF9316B	reserved	
– 0xFFF9316C	Inbound Message 1 Frame Queue Dequeue Pointer Address Register	IM1FQDPAR
– 0xFFF93170–0xFFF93173	reserved	
– 0xFFF93174	Inbound Message 1 Frame Queue Enqueue Pointer Address Register	IM1FQEPAR
– 0xFFF93178	Inbound Message 1 Maximum Interrupt Report Interval Register	IM1MIRIR
– 0xFFF9317C–0xFFF933FF	reserved	
– 0xFFF93400	Outbound Doorbell Mode Register	ODMR
– 0xFFF93404	Outbound Doorbell Status Register	ODSR
– 0xFFF93408–0xFFF93417	reserved	
– 0xFFF93418	Outbound Doorbell Destination Port Register	ODDPR
– 0xFFF9341C	Outbound Doorbell Destination Attributes Register	ODDATR
– 0xFFF93420–0xFFF9342B	reserved	
– 0xFFF9342C	Outbound Doorbell Retry Error Threshold Configuration Register	ODRETCR
– 0xFFF93430–0xFFF9345F	reserved	
– 0xFFF93460	Inbound Doorbell Mode Register	IDMR
– 0xFFF93464	Inbound Doorbell Status Register	IDSR
– 0xFFF93468–0xFFF9346B	reserved	
– 0xFFF9346C	Inbound Doorbell Queue Dequeue Pointer Address Register	IDQDPAR
– 0xFFF93470–0xFFF93473	reserved	
– 0xFFF93474	Inbound Doorbell Queue Enqueue Pointer Address Register	IDQEPAR
– 0xFFF93478	Inbound Doorbell Maximum Interrupt Report Interval Register	IDMIRIR
– 0xFFF9347C–0xFFF934DF	reserved	
– 0xFFF934E0	Inbound Port-write Mode Register	IPWMR
– 0xFFF934E4	Inbound Port-write Status Register	IPWSR
– 0xFFF934EC	Inbound Port-write Queue Base Address Register	IPWQBAR
– 0xFFF934F0–0xFFF9FFFF	reserved	
• 0xFFFA0000–0xFFFA0FFF	reserved	
• 0xFFFA1000–0xFFFA103F	HSSI OCN Crossbar Switch to MBus0 (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*)	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFA1000	OCN-to-MBus Control Register 0	O2MCR0
• 0xFFFA1040–0xFFFA107F	HSSI OCN Crossbar Switch to MBus1 (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*)	
– 0xFFFA1040	OCN-to-MBus Control Register 1	O2MCR1
– 0xFFA1044–0xFFFA107F	reserved	
• 0xFFFA1080–0xFFFA7FFF	reserved	
• 0xFFFA8000–0xFFFA8FFF	HSSI Dedicated DMA Controller 0 Registers (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*)	
– 0xFFFA8000–0xFFFA80FF	reserved	
– 0xFFFA8100	DMA 0 Mode Register	D0MR0
– 0xFFFA8104	DMA 0 Status Register	D0SR0
– 0xFFFA8108	DMA 0 Current Link Descriptor Extended Address Register	D0ECLNDAR0
– 0xFFFA810C	DMA 0 Current Link Descriptor Address Register	D0CLNDAR0
– 0xFFFA8110	DMA 0 Source Attributes Register	D0SATR0
– 0xFFFA8114	DMA 0 Source Address Register	D0SAR0
– 0xFFFA8118	DMA 0 Destination Attributes Register	D0DATR0
– 0xFFFA811C	DMA 0 Destination Address Register	D0DAR0
– 0xFFFA8120	DMA 0 Byte Count Register	D0BCR0
– 0xFFFA8124–0xFFFA8127	reserved	
– 0xFFFA8128	DMA 0 Next Link Descriptor Address Register	D0NLNDAR0
– 0xFFFA812C–0xFFFA8133	reserved	
– 0xFFFA8134	DMA 0 Current List Descriptor Address Register	D0CLSDAR0
– 0xFFFA8138	DMA 0 Next List Descriptor Extended Address Register	D0ENLSDAR0
– 0xFFFA813C	DMA 0 Next List Descriptor Address Register	D0NLSDAR0
– 0xFFFA8140	DMA 0 Source Stride Register	D0SSR0
– 0xFFFA8144	DMA 0 Destination Stride Register	D0DSR0
– 0xFFFA8148–0xFFFA817F	reserved	
– 0xFFFA8180	DMA 1 Mode Register	D0MR1
– 0xFFFA8184	DMA 1 Status Register	D0SR1
– 0xFFFA8188	DMA 1 Current Link Descriptor Extended Address Register	D0ECLNDAR1
– 0xFFFA818C	DMA 1 Current Link Descriptor Address Register	D0CLNDAR1
– 0xFFFA8190	DMA 1 Source Attributes Register	D0SATR1
– 0xFFFA8194	DMA 1 Source Address Register	D0SAR1
– 0xFFFA8198	DMA 1 Destination Attributes Register	D0DATR1
– 0xFFFA819C	DMA 1 Destination Address Register	D0DAR1
– 0xFFFA81A0	DMA 1 Byte Count Register	D0BCR1
– 0xFFFA81A4	DMA 1 Next Link Descriptor Extended Address Register	D0ENLNDAR1
– 0xFFFA81A8	DMA 1 Next Link Descriptor Address Register	D0NLNDAR1

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFA81AC–0xFFFA81AF	reserved	
– 0xFFFA81B0	DMA 1 Current List Descriptor Extended Address Register	D0ECLSDAR1
– 0xFFFA81B4	DMA 1 Current List Descriptor Address Register	D0CLSDAR1
– 0xFFFA81B8	DMA 1 Next List Descriptor Extended Address Register	D0ENLSDAR1
– 0xFFFA81BC	DMA 1 Next List Descriptor Address Register	D0NLSDAR1
– 0xFFFA81C0	DMA 1 Source Stride Register	D0SSR1
– 0xFFFA81C4	DMA 1 Destination Stride Register	D0DSR1
– 0xFFFA81C8–0xFFFA81FF	reserved	
– 0xFFFA8200	DMA 2 Mode Register	D0MR2
– 0xFFFA8204	DMA 2 Status Register	D0SR2
– 0xFFFA8208	DMA 2 Current Link Descriptor Extended Address Register	D0ECLNDAR2
– 0xFFFA820C	DMA 2 Current Link Descriptor Address Register	D0CLNDAR2
– 0xFFFA8210	DMA 2 Source Attributes Register	D0SATR2
– 0xFFFA8214	DMA 2 Source Address Register	D0SAR2
– 0xFFFA8218	DMA 2 Destination Attributes Register	D0DATR2
– 0xFFFA821C	DMA 2 Destination Address Register	D0DAR2
– 0xFFFA8220	DMA 2 Byte Count Register	D0BCR2
– 0xFFFA8224	DMA 2 Next Link Descriptor Extended Address Register	D0ENLNDAR2
– 0xFFFA8228	DMA 2 Next Link Descriptor Address Register	D0NLNDAR2
– 0xFFFA822C–0xFFFA822F	reserved	
– 0xFFFA8230	DMA 2 Current List Descriptor Extended Address Register	D0ECLSDAR2
– 0xFFFA8234	DMA 2 Current List Descriptor Address Register	D0CLSDAR2
– 0xFFFA8238	DMA 2 Next List Descriptor Extended Address Register	D0ENLSDAR2
– 0xFFFA823C	DMA 2 Next List Descriptor Address Register	D0NLSDAR2
– 0xFFFA8240	DMA 2 Source Stride Register	D0SSR2
– 0xFFFA8244	DMA 2 Destination Stride Register	D0DSR2
– 0xFFFA8248–0xFFFA827F	reserved	
– 0xFFFA8280	DMA 3 Mode Register	D0MR3
– 0xFFFA8284	DMA 3 Status Register	D0SR3
– 0xFFFA8288	DMA 3 Current Link Descriptor Extended Address Register	D0ECLNDAR3
– 0xFFFA828C	DMA 3 Current Link Descriptor Address Register	D0CLNDAR3
– 0xFFFA8290	DMA 3 Source Attributes Register	D0SATR3
– 0xFFFA8294	DMA 3 Source Address Register	D0SAR3
– 0xFFFA8298	DMA 3 Destination Attributes Register	D0DATR3
– 0xFFFA829C	DMA 3 Destination Address Register	D0DAR3
– 0xFFFA82A0	DMA 3 Byte Count Register	D0BCR3
– 0xFFFA82A4	DMA 3 Next Link Descriptor Extended Address Register	D0ENLNDAR3
– 0xFFFA82A8	DMA 3 Next Link Descriptor Address Register	D0NLNDAR3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFA82AC–0xFFFA82AF	reserved	
– 0xFFFA82B0	DMA 3 Current List Descriptor Extended Address Register	D0ECLSDAR3
– 0xFFFA82B4	DMA 3 Current List Descriptor Address Register	D0CLSDAR3
– 0xFFFA82B8	DMA 3 Next List Descriptor Extended Address Register	D0ENLSDAR3
– 0xFFFA82BC	DMA 3 Next List Descriptor Address Register	D0NLSDAR3
– 0xFFFA82C0	DMA 3 Source Stride Register	D0SSR3
– 0xFFFA82C4	DMA 3 Destination Stride Register	D0DSR3
– 0xFFFA82C8–0xFFFA82FF	reserved	
– 0xFFFA8300	DMA General Status Register	D0DGSR
– 0xFFFA8304–0xFFFA9C07	reserved	
– 0xFFFA9C08	Local Access Window Base Address Register 0	D0LAWBAR0
– 0xFFFA9C0C–0xFFFA9C0F	reserved	
– 0xFFFA9C10	Local Access Window Attributes Register 0	D0LAWAR0
– 0xFFFA9C14–0xFFFA9C27	reserved	
– 0xFFFA9C28	Local Access Window Base Address Register 1	D0LAWBAR1
– 0xFFFA9C2C–0xFFFA9C2F	reserved	
– 0xFFFA9C30	Local Access Window Attributes Register 1	D0LAWAR1
– 0xFFFA9C34–0xFFFA9C47	reserved	
– 0xFFFA9C48	Local Access Window Base Address Register 2	D0LAWBAR2
– 0xFFFA9C4C–0xFFFA9C4F	reserved	
– 0xFFFA9C50	Local Access Window Attributes Register 2	D0LAWAR2
– 0xFFFA9C54–0xFFFA9C67	reserved	
– 0xFFFA9C68	Local Access Window Base Address Register 3	D0LAWBAR3
– 0xFFFA9C6C–0xFFFA9C6F	reserved	
– 0xFFFA9C70	Local Access Window Attributes Register 3	D0LAWAR3
– 0xFFFA9C74–0xFFFA9C87	reserved	
– 0xFFFA9C88	Local Access Window Base Address Register 4	D0LAWBAR4
– 0xFFFA9C8C–0xFFFA9C8F	reserved	
– 0xFFFA9C90	Local Access Window Attributes Register 4	D0LAWAR4
– 0xFFFA9C94–0xFFFA9CA7	reserved	
– 0xFFFA9CA8	Local Access Window Base Address Register 5	D0LAWBAR5

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFA9CAC–0xFFFA9CAF	reserved	
– 0xFFFA9CB0	Local Access Window Attributes Register 5	D0LAWAR5
– 0xFFFA9CB4–0xFFFA9CC7	reserved	
– 0xFFFA9CC8	Local Access Window Base Address Register 6	D0LAWBAR6
– 0xFFFA9CCC–0xFFFA9CCF	reserved	
– 0xFFFA9CD0	Local Access Window Attributes Register 6	D0LAWAR6
– 0xFFFA9CD4–0xFFFA9CE7	reserved	
– 0xFFFA9CE8	Local Access Window Base Address Register 7	D0LAWBAR7
– 0xFFFA9CEC–0xFFFA9CEF	reserved	
– 0xFFFA9CF0	Local Access Window Attributes Register 7	D0LAWAR7
– 0xFFFA9CF4–0xFFFA9D07	reserved	
– 0xFFFA9D08	Local Access Window Base Address Register 8	D0LAWBAR8
– 0xFFFA9D0C–0xFFFA9D0F	reserved	
– 0xFFFA9D10	Local Access Window Attributes Register 8	D0LAWAR8
– 0xFFFA9D14–0xFFFA9D27	reserved	
– 0xFFFA9D28	Local Access Window Base Address Register 9	D0LAWBAR9
– 0xFFFA9D2C–0xFFFA9D2F	reserved	
– 0xFFFA9D30	Local Access Window Attributes Register 9	D0LAWAR9
– 0xFFFA9D34–0xFFFA8FFF	reserved	
• 0xFFFA9000–0xFFFA9FFF	DMA Controller 0 to OCN	
• 0xFFFAA000–0xFFFAAFFF	Dedicated DMA Controller 1 Registers (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*)	
– 0xFFFAA000–0xFFFAA0FF	reserved	
– 0xFFFAA100	DMA 0 Mode Register	D1MR0
– 0xFFFAA104	DMA 0 Status Register	D1SR0
– 0xFFFAA108	DMA 0 Current Link Descriptor Extended Address Register	D1ECLNDAR0
– 0xFFFAA10C	DMA 0 Current Link Descriptor Address Register	D1CLNDAR0
– 0xFFFAA110	DMA 0 Source Attributes Register	D1SATR0
– 0xFFFAA114	DMA 0 Source Address Register	D1SAR0
– 0xFFFAA118	DMA 0 Destination Attributes Register	D1DATR0
– 0xFFFAA11C	DMA 0 Destination Address Register	D1DAR0
– 0xFFFAA120	DMA 0 Byte Count Register	D1BCR0

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFAA124–0xFFFAA127	reserved	
– 0xFFFAA128	DMA 0 Next Link Descriptor Address Register	D1NLNDAR0
– 0xFFFAA12C–0xFFFAA133	reserved	
– 0xFFFAA134	DMA 0 Current List Descriptor Address Register	D1CLSDAR0
– 0xFFFAA138	DMA 0 Next List Descriptor Extended Address Register	D1ENLSDAR0
– 0xFFFAA13C	DMA 0 Next List Descriptor Address Register	D1NLSDAR0
– 0xFFFAA140	DMA 0 Source Stride Register	D1SSR0
– 0xFFFAA144	DMA 0 Destination Stride Register	D1DSR0
– 0xFFFAA148–0xFFFAA17F	reserved	
– 0xFFFAA180	DMA 1 Mode Register	D1MR1
– 0xFFFAA184	DMA 1 Status Register	D1SR1
– 0xFFFAA188	DMA 1 Current Link Descriptor Extended Address Register	D1ECLNDAR1
– 0xFFFAA18C	DMA 1 Current Link Descriptor Address Register	D1CLNDAR1
– 0xFFFAA190	DMA 1 Source Attributes Register	D1SATR1
– 0xFFFAA194	DMA 1 Source Address Register	D1SAR1
– 0xFFFAA198	DMA 1 Destination Attributes Register	D1DATR1
– 0xFFFAA19C	DMA 1 Destination Address Register	D1DAR1
– 0xFFFAA1A0	DMA 1 Byte Count Register	D1BCR1
– 0xFFFAA1A4	DMA 1 Next Link Descriptor Extended Address Register	D1ENLNDAR1
– 0xFFFAA1A8	DMA 1 Next Link Descriptor Address Register	D1NLNDAR1
– 0xFFFAA1AC–0xFFFAA1AF	reserved	
– 0xFFFAA1B0	DMA 1 Current List Descriptor Extended Address Register	D1ECLSDAR1
– 0xFFFAA1B4	DMA 1 Current List Descriptor Address Register	D1CLSDAR1
– 0xFFFAA1B8	DMA 1 Next List Descriptor Extended Address Register	D1ENLSDAR1
– 0xFFFAA1BC	DMA 1 Next List Descriptor Address Register	D1NLSDAR1
– 0xFFFAA1C0	DMA 1 Source Stride Register	D1SSR1
– 0xFFFAA1C4	DMA 1 Destination Stride Register	D1DSR1
– 0xFFFAA1C8–0xFFFAA1FF	reserved	
– 0xFFFAA200	DMA 2 Mode Register	D1MR2
– 0xFFFAA204	DMA 2 Status Register	D1SR2
– 0xFFFAA208	DMA 2 Current Link Descriptor Extended Address Register	D1ECLNDAR2
– 0xFFFAA20C	DMA 2 Current Link Descriptor Address Register	D1CLNDAR2
– 0xFFFAA210	DMA 2 Source Attributes Register	D1SATR2
– 0xFFFAA214	DMA 2 Source Address Register	D1SAR2
– 0xFFFAA218	DMA 2 Destination Attributes Register	D1DATR2
– 0xFFFAA21C	DMA 2 Destination Address Register	D1DAR2
– 0xFFFAA220	DMA 2 Byte Count Register	D1BCR2
– 0xFFFAA224	DMA 2 Next Link Descriptor Extended Address Register	D1ENLNDAR2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFAA228	DMA 2 Next Link Descriptor Address Register	D1NLNDAR2
– 0xFFFAA22C–0xFFFAA22F	reserved	
– 0xFFFAA230	DMA 2 Current List Descriptor Extended Address Register	D1ECLSDAR2
– 0xFFFAA234	DMA 2 Current List Descriptor Address Register	D1CLSDAR2
– 0xFFFAA238	DMA 2 Next List Descriptor Extended Address Register	D1ENLSDAR2
– 0xFFFAA23C	DMA 2 Next List Descriptor Address Register	D1NLSDAR2
– 0xFFFAA240	DMA 2 Source Stride Register	D1SSR2
– 0xFFFAA244	DMA2 Destination Stride Register	D1DSR2
– 0xFFFAA248–0xFFFAA27F	reserved	
– 0xFFFAA280	DMA 3 Mode Register	D1MR3
– 0xFFFAA284	DMA 3 Status Register	D1SR3
– 0xFFFAA288	DMA 3 Current Link Descriptor Extended Address Register	D1ECLNDAR3
– 0xFFFAA28C	DMA 3 Current Link Descriptor Address Register	D1CLNDAR3
– 0xFFFAA290	DMA 3 Source Attributes Register	D1SATR3
– 0xFFFAA294	DMA 3 Source Address Register	D1SAR3
– 0xFFFAA298	DMA 3 Destination Attributes Register	D1DATR3
– 0xFFFAA29C	DMA 3 Destination Address Register	D1DAR3
– 0xFFFAA2A0	DMA 3 Byte Count Register	D1BCR3
– 0xFFFAA2A4	DMA 3 Next Link Descriptor Extended Address Register	D1ENLNDAR3
– 0xFFFAA2A8	DMA 3 Next Link Descriptor Address Register	D1NLNDAR3
– 0xFFFAA2AC–0xFFFAA2AF	reserved	
– 0xFFFAA2B0	DMA 3 Current List Descriptor Extended Address Register	D1ECLSDAR3
– 0xFFFAA2B4	DMA 3 Current List Descriptor Address Register	D1CLSDAR3
– 0xFFFAA2B8	DMA 3 Next List Descriptor Extended Address Register	D1ENLSDAR3
– 0xFFFAA2BC	DMA 3 Next List Descriptor Address Register	D1NLSDAR3
– 0xFFFAA2C0	DMA 3 Source Stride Register	D1SSR3
– 0xFFFAA2C4	DMA 3 Destination Stride Register	D1DSR3
– 0xFFFAA2C8–0xFFFAA2FF	reserved	
– 0xFFFAA300	DMA General Status Register	D1DGSR
– 0xFFFAA304–0xFFFABC07	reserved	
– 0xFFFABC08	Local Access Window Base Address Register 0	D1LAWBAR0
– 0xFFFABC0C–0xFFFABC0F	reserved	
– 0xFFFABC10	Local Access Window Attributes Register 0	D1LAWAR0
– 0xFFFABC14–0xFFFABC27	reserved	
– 0xFFFABC28	Local Access Window Base Address Register 1	D1LAWBAR1
– 0xFFFABC2C–0xFFFABC2F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFABC30	Local Access Window Attributes Register 1	D1LAWAR1
– 0xFFFABC34–0xFFFABC47	reserved	
– 0xFFFABC48	Local Access Window Base Address Register 2	D1LAWBAR2
– 0xFFFABC4C–0xFFFABC4F	reserved	
– 0xFFFABC50	Local Access Window Attributes Register 2	D1LAWAR2
– 0xFFFABC54–0xFFFABC67	reserved	
– 0xFFFABC68	Local Access Window Base Address Register 3	D1LAWBAR3
– 0xFFFABC6C–0xFFFABC6F	reserved	
– 0xFFFABC70	Local Access Window Attributes Register 3	D1LAWAR3
– 0xFFFABC74–0xFFFABC87	reserved	
– 0xFFFABC88	Local Access Window Base Address Register 4	D1LAWBAR4
– 0xFFFABC8C–0xFFFABC8F	reserved	
– 0xFFFABC90	Local Access Window Attributes Register 4	D1LAWAR4
– 0xFFFABC94–0xFFFABCA7	reserved	
– 0xFFFABCA8	Local Access Window Base Address Register 5	D1LAWBAR5
– 0xFFFABCAC–0xFFFABCAF	reserved	
– 0xFFFABCB0	Local Access Window Attributes Register 5	D1LAWAR5
– 0xFFFABCB4–0xFFFABCC7	reserved	
– 0xFFFABCC8	Local Access Window Base Address Register 6	D1LAWBAR6
– 0xFFFABCCC–0xFFFABCCF	reserved	
– 0xFFFABCD0	Local Access Window Attributes Register 6	D1LAWAR6
– 0xFFFABCD4–0xFFFABCE7	reserved	
– 0xFFFABCE8	Local Access Window Base Address Register 7	D1LAWBAR7
– 0xFFFABCEC–0xFFFABCEF	reserved	
– 0xFFFABCF0	Local Access Window Attributes Register 7	D1LAWAR7
– 0xFFFABCF4–0xFFFABD07	reserved	
– 0xFFFABD08	Local Access Window Base Address Register 8	D1LAWBAR8
– 0xFFFABD0C–0xFFFABD0F	reserved	
– 0xFFFABD10	Local Access Window Attributes Register 8	D1LAWAR8
– 0xFFFABD14–0xFFFABD27	reserved	
– 0xFFFABD28	Local Access Window Base Address Register 9	D1LAWBAR9

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFABD2C–0xFFFABD2F	reserved	
– 0xFFFABD30	Local Access Window Attributes Register 9	D1LAWAR9
– 0xFFFABD34–0xFFFA8FFF	reserved	
• 0xFFFAB000–0xFFFABFFF	DMA Controller 1 to OCN	
• 0xFFFAC000–0xFFFAC01F	SerDes PHY 1 Registers	
• 0xFFFAC000	SRDS Control Register 0 for SerDes Port 1	SRDS1CR0
• 0xFFFAC004	SRDS Control Register 1 for SerDes Port 1	SRDS1CR1
• 0xFFFAC008	SRDS Control Register 2 for SerDes Port 1	SRDS1CR2
• 0xFFFAC00C	SRDS Control Register 3 for SerDes Port 1	SRDS1CR3
• 0xFFFAC010	SRDS Control Register 4 for SerDes Port 1	SRDS1CR4
• 0xFFFAC014	SRDS Control Register 5 for SerDes Port 1	SRDS1CR5
• 0xFFFAC018	SRDS Control Register 6 for SerDes Port 1	SRDS1CR6
• 0xFFFAC0020–0xFFFACFFF	reserved	
• 0xFFFAD000–0xFFFAD01F	SerDes PHY 2 Registers	
• 0xFFFAD000	SRDS Control Register 0 for SerDes Port 2	SRDS2CR0
• 0xFFFAD004	SRDS Control Register 1 for SerDes Port 2	SRDS2CR1
• 0xFFFAD008	SRDS Control Register 2 for SerDes Port 2	SRDS2CR2
• 0xFFFAD00C	SRDS Control Register 3 for SerDes Port 2	SRDS2CR3
• 0xFFFAD010	SRDS Control Register 4 for SerDes Port 2	SRDS2CR4
• 0xFFFAD014	SRDS Control Register 5 for SerDes Port 2	SRDS2CR5
• 0xFFFAD018	SRDS Control Register 6 for SerDes Port 2	SRDS2CR6
• 0xFFFAD020–0xFFFB6FFF	reserved	
• 0xFFFB7000–0xFFFB7FFF	PCI Express Registers (see **Chapter 17**, *PCI Express Controller*)	
– 0xFFFB7000	PCI Express Configuration Address Register	PEX_CONFIG_ADDR
– 0xFFFB7004	PCI Express Configuration Data Register	PEX_CONFIG_DATA
– 0xFFFB7008–0xFFFB700B	reserved	
– 0xFFFB700C	PCI Express Outbound Completion Timeout Register	PEX_OTB_CPL_TOR
– 0xFFFB7010	PCI Express Configuration Retry Timeout Register	PEX_CONF_RTY_TOR
– 0xFFFB7014	PCI Express Configuration Register	PEX_CONFIG
– 0xFFFB7018–0xFFFB701F	reserved	
– 0xFFFB7020	PCI Express PME and Message Detect Register	PEX_PME_MES_DR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFB7024	PCI Express PME and Message Disable Register	PEX_PME_MES_DISR
– 0xFFFB7028	PCI Express PME and Message Interrupt Enable Register	PEX_PME_MES_IER
– 0xFFFB702C	PCI Express Power Management Command Register	PEX_PMCR
– 0xFFFB7030–0xFFFB70FF	reserved	
– 0xFFFB7100	PCI Express Link Width Control Register	PEX_LWCR
– 0xFFFB7104	PCI Express Link Width Status Register	PEX_LWSR
– 0xFFFB7108	PCI Express Link Speed Control Register	PEX_LSCR
– 0xFFFB710C	PCI Express Link Speed Status Register	PEX_LSSR
– 0xFFFB7110–0xFFFB7BF7	reserved	
– 0xFFFB7BF8	IP Block Revision Register 1	PEX_IP_BLK_REV1
– 0xFFFB7BFC	IP Block Revision Register 2	PEX_IP_BLK_REV2
– 0xFFFB7C00	PCI Express Outbound Translation Address Register 0	PEXOTAR0
– 0xFFFB7C04	PCI Express Outbound Translation Extended Address Register 0	PEXOTEAR0
– 0xFFFB7C08–0xFFFB7C0F	reserved	
– 0xFFFB7C10	PCI Express Outbound Window Attributes Register 0	PEXOWAR0
– 0xFFFB7C14–0xFFFB7C1F	reserved	
– 0xFFFB7C20	PCI Express Outbound Translation Address Register 1	PEXOTAR1
– 0xFFFB7C24	PCI Express Outbound Translation Extended Address Register 1	PEXOTEAR1
– 0xFFFB7C28	PCI Express Outbound Window Base Address Register 1	PEXOWBAR1
– 0xFFFB7C2C–0xFFFB7C2F	reserved	
– 0xFFFB7C30	PCI Express Outbound Window Attributes Register 1	PEXOWAR1
– 0xFFFB7C34–0xFFFB7C3F	reserved	
– 0xFFFB7C40	PCI Express Outbound Translation Address Register 2	PEXOTAR2
– 0xFFFB7C44	PCI Express Outbound Translation Extended Address Register 2	PEXOTEAR2
– 0xFFFB7C48	PCI Express Outbound Window Base Address Register 2	PEXOWBAR2
– 0xFFFB7C4C–0xFFFB7C4F	reserved	
– 0xFFFB7C50	PCI Express Outbound Window Attributes Register 2	PEXOWAR2
– 0xFFFB7C54–0xFFFB7C5F	reserved	
– 0xFFFB7C60	PCI Express Outbound Translation Address Register 3	PEXOTAR3
– 0xFFFB7C64	PCI Express Outbound Translation Extended Address Register 3	PEXOTEAR3
– 0xFFFB7C68	PCI Express Outbound Window Base Address Register 3	PEXOWBAR3
– 0xFFFB7C6C–0xFFFB7C6F	reserved	
– 0xFFFB7C70	PCI Express Outbound Window Attributes Register 3	PEXOWAR3

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFB7C74–0xFFFB7C7F	reserved	
– 0xFFFB7C80	PCI Express Outbound Translation Address Register 4	PEXOTAR4
– 0xFFFB7C84	PCI Express Outbound Translation Extended Address Register 4	PEXOTEAR4
– 0xFFFB7C88	PCI Express Outbound Window Base Address Register 4	PEXOWBAR4
– 0xFFFB7C8C–0xFFFB7C8F	reserved	
– 0xFFFB7C90	PCI Express Outbound Window Attributes Register 4	PEXOWAR4
– 0xFFFB7C94–0xFFFB7D9F	reserved	
– 0xFFFB7DA0	PCI Express Inbound Translation Address Register 3	PEXITAR3
– 0xFFFB7DA4–0xFFFB7DA7	reserved	
– 0xFFFB7DA8	PCI Express Inbound Window Base Address Register 3	PEXIWBAR3
– 0xFFFB7DAC	PCI Express Inbound Window Base Extended Address Register 3	PEXIWBEAR3
– 0xFFFB7DB0	PCI Express Inbound Window Attributes Register 3	PEXIWAR3
– 0xFFFB7DB4–0xFFFB7DBF	reserved	
– 0xFFFB7DC0	PCI Express Inbound Translation Address Register 2	PEXITAR2
– 0xFFFB7DC4–0xFFFB7DC7	reserved	
– 0xFFFB7DC8	PCI Express Inbound Window Base Address Register 2	PEXIWBAR2
– 0xFFFB7DCC	PCI Express Inbound Window Base Extended Address Register 2	PEXIWBEAR2
– 0xFFFB7DD0	PCI Express Inbound Window Attributes Register 2	PEXIWAR2
– 0xFFFB7DD4–0xFFFB7DDF	reserved	
– 0xFFFB7DE0	PCI Express Inbound Translation Address Register 1	PEXITAR1
– 0xFFFB7DE4–0xFFFB7DE7	reserved	
– 0xFFFB7DE8	PCI Express Inbound Window Base Address Register 1	PEXIWBAR1
– 0xFFFB7DEC–0xFFFB7DEF	reserved	
– 0xFFFB7DF0	PCI Express Inbound Window Attributes Register 1	PEXIWAR1
– 0xFFFB7DF4–0xFFFB7DFF	reserved	
– 0xFFFB7E00	PCI Express Error Detect Register	PEX_ERR_DR
– 0xFFFB7E04–0xFFFB7E07	reserved	
– 0xFFFB7E08	PCI Express Error Interrupt Enable Register	PEX_ERR_EN
– 0xFFFB7E0C–0xFFFB7E0F	reserved	
– 0xFFFB7E10	PCI Express Error Disable Register	PEX_ERR_DISR
– 0xFFFB7E14–0xFFFB7E1F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFB7E20	PCI Express Error Capture Status Register	PEX_ERR_CAP_STAT
– 0xFFFB7E24–0xFFFB7E27	reserved	
– 0xFFFB7E28	PCI Express Error Capture Register 0	PEX_ERR_CAP_R0
– 0xFFFB7E2C	PCI Express Error Capture Register 1	PEX_ERR_CAP_R1
– 0xFFFB7E30	PCI Express Error Capture Register 2	PEX_ERR_CAP_R2
– 0xFFFB7E34	PCI Express Error Capture Register 3	PEX_ERR_CAP_R3
– 0xFFFB7E38–0x FFFB7FFF	reserved	
• 0xFFFBB000–0xFFFBB7FF	RapidIO/PCI Express Security Bridge Registers	
• 0xFFFBB800–0xFFFBB8FF	Performance Monitor	
– 0xFFFBB800	Performance Monitor Global Control Register	PMGCR
– 0xFFFBB804–0xFFFBB80F	reserved	
– 0xFFFBB810	Performance Monitor Local Control Register A0	PMLCA0
– 0xFFFBB814	reserved	
– 0xFFFBB818	Performance Monitor Counter 0	PMC0
– 0xFFFBB81C–0xFFFBB81F	reserved	
– 0xFFFBB820	Performance Monitor Local Control Register A1	PMLCA1
– 0xFFFBB824	Performance Monitor Local Control Register B1	PMLCB1
– 0xFFFBB828	Performance Monitor Counter 1	PMC1
– 0xFFFBB82C–0xFFFBB82F	reserved	
– 0xFFFBB830	Performance Monitor Local Control Register A2	PMLCA2
– 0xFFFBB834	Performance Monitor Local Control Register B2	PMLCB2
– 0xFFFBB838	Performance Monitor Counter 2	PMC2
– 0xFFFBB83C–0xFFFBB83F	reserved	
– 0xFFFBB840	Performance Monitor Local Control Register A3	PMLCA3
– 0xFFFBB844	Performance Monitor Local Control Register B3	PMLCB3
– 0xFFFBB848	Performance Monitor Counter 3	PMC3
– 0xFFFBB84C–0xFFFBB84F	reserved	
– 0xFFFBB850	Performance Monitor Local Control Register A4	PMLCA4
– 0xFFFBB854	Performance Monitor Local Control Register B4	PMLCB4
– 0xFFFBB858	Performance Monitor Counter 4	PMC4
– 0xFFFBB85C–0xFFFBB85F	reserved	
– 0xFFFBB860	Performance Monitor Local Control Register A5	PMLCA5
– 0xFFFBB864	Performance Monitor Local Control Register B5	PMLCB5
– 0xFFFBB868	Performance Monitor Counter 5	PMC5

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFBB86C–0xFFFBB86F	reserved	
– 0xFFFBB870	Performance Monitor Local Control Register A6	PMLCA6
– 0xFFFBB874	Performance Monitor Local Control Register B6	PMLCB6
– 0xFFFBB878	Performance Monitor Counter 6	PMC6
– 0xFFFBB87C–0xFFFBB87F	reserved	
– 0xFFFBB880	Performance Monitor Local Control Register A7	PMLCA7
– 0xFFFBB884	Performance Monitor Local Control Register B7	PMLCB7
– 0xFFFBB888	Performance Monitor Counter 7	PMC7
– 0xFFFBB88C–0xFFFBB88F	reserved	
– 0xFFFBB890	Performance Monitor Local Control Register A8	PMLCA8
– 0xFFFBB894	Performance Monitor Local Control Register B8	PMLCB8
– 0xFFFBB898	Performance Monitor Counter 8	PMC8
– 0xFFFBB89C–0xFFFBB8FF	reserved	
• 0xFFFBB900–0xFFFCFFFF	reserved	
• 0xFFFD0000–0xFFFDFFFF	Security Engine Channel 1–4 Secondary Addresses (see **Chapter 27**, *Security Engine (SEC)*)	
– 0xFFFD0000–0xFFFD00FF	reserved	
– 0xFFFD0100–0xFFFD01FF	SEC Channel 1 Secondary Addresses (used if MCR[RCA1] = 1)	
– 0xFFFD0100–0xFFFD0107	reserved	
– 0xFFFD0108	Channel 1 Configuration Register	CCR1
– 0xFFFD0110	Channel 1 Status Register	CSR1
– 0xFFFD0118–0xFFFD013F	reserved	
– 0xFFFD0140	Channel 1 Current Descriptor Pointer Register	CDPR1
– 0xFFFD0148	Channel 1 Fetch FIFO	CFF1
– 0xFFFD0150–0xFFFD017F	reserved	
– 0xFFFD0180–0xFFFD01BF	Channel 1 Descriptor Buffer	DB1
– 0xFFFD01C0–0xFFFD01DF	Channel 1 Gather Link Tables	—
– 0xFFFD01E0–0xFFFD01FF	Channel 1 Scatter Link Tables	—
– 0xFFFD0200–0xFFFD02FF	SEC Channel 2 Secondary Addresses (used if MCR[RCA2] = 1)	
– 0xFFFD0200–0xFFFD0207	reserved	
– 0xFFFD0208	Channel 2 Configuration Register	CCR2

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD0210	Channel 2 Status Register	CSR2
– 0xFFFD0218–0xFFFD023F	reserved	
– 0xFFFD0240	Channel 2 Current Descriptor Pointer Register	CDPR2
– 0xFFFD0248	Channel 2 Fetch FIFO	CFF2
– 0xFFFD0250–0xFFFD027F	reserved	
– 0xFFFD0280–0xFFFD02BF	Channel 2 Descriptor Buffer	DB2
– 0xFFFD02C0–0xFFFD02DF	Channel 2 Gather Link Tables	—
– 0xFFFD02E0–0xFFFD02FF	Channel 2 Scatter Link Tables	—
– 0xFFFD0300–0xFFFD03FF	SEC Channel 3 Secondary Addresses (used if MCR[RCA3] = 1)	
– 0xFFFD0300–0xFFFD0307	reserved	
– 0xFFFD0308	Channel 3 Configuration Register	CCR3
– 0xFFFD0310	Channel 3 Status Register	CSR3
– 0xFFFD0318–0xFFFD033F	reserved	
– 0xFFFD0340	Channel 3 Current Descriptor Pointer Register	CDPR3
– 0xFFFD0348	Channel 3 Fetch FIFO	CFF3
– 0xFFFD0350–0xFFFD037F	reserved	
– 0xFFFD0380–0xFFFD03BF	Channel 3 Descriptor Buffer	DB3
– 0xFFFD03C0–0xFFFD03DF	Channel 3 Gather Link Tables	—
– 0xFFFD03E0–0xFFFD03FF	Channel 3 Scatter Link Tables	—
– 0xFFFD0400–0xFFFD04FF	SEC Channel 4 Secondary Addresses (used if MCR[RCA4] = 1)	
– 0xFFFD0400–0xFFFD0407	reserved	
– 0xFFFD0408	Channel 4 Configuration Register	CCR4
– 0xFFFD0410	Channel 4 Status Register	CSR4
– 0xFFFD0418–0xFFFD043F	reserved	
– 0xFFFD0440	Channel 4 Current Descriptor Pointer Register	CDPR4
– 0xFFFD0448	Channel 4 Fetch FIFO	CFF4
– 0xFFFD0450–0xFFFD047F	reserved	
– 0xFFFD0480–0xFFFD04BF	Channel 4 Descriptor Buffer	DB4

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD04C0–0xFFFD04DF	Channel 4 Gather Link Tables	—
– 0xFFFD04E0–0xFFFD04FF	Channel 4 Scatter Link Tables	—
– 0xFFFD0400–0xFFFD04FF	reserved	
– 0xFFFD0500–0xFFFD0FFF	reserved	
– 0xFFFD1000–0xFFFD10FF	SEC Controller	
– 0xFFFD1000–0xFFFD1007	reserved	
– 0xFFFD1008	Controller Interrupt Enable Register	CIER
– 0xFFFD1010	Controller Interrupt Status Register	CISR
– 0xFFFD1018	Controller Interrupt Clear Register	CICR
– 0xFFFD1020	Controller Identification Register	CIDR
– 0xFFFD1028	EU Assignment Status Register	EUASR
– 0xFFFD1030	Master Control Register	MCR
– 0xFFFD1038–0xFFFD10FF	reserved	
– 0xFFFD1100–0xFFFD11FF	SEC Channel 1 Primary Addresses (used if MCR[RCA1] = 0)	
– 0xFFFD1100–0xFFFD1107	reserved	
– 0xFFFD1108	Channel 1 Configuration Register	CCR1
– 0xFFFD1110	Channel 1 Status Register	CSR1
– 0xFFFD1118–0xFFFD113F	reserved	
– 0xFFFD1140	Channel 1 Current Descriptor Pointer Register	CDPR1
– 0xFFFD1148	Channel 1 Fetch FIFO	CFF1
– 0xFFFD1150–0xFFFD117F	reserved	
– 0xFFFD1180–0xFFFD11BF	Channel 1 Descriptor Buffer	DB1
– 0xFFFD11C0–0xFFFD11DF	Channel 1 Gather Link Tables	—
– 0xFFFD11E0–0xFFFD11FF	Channel 1 Scatter Link Tables	—
– 0xFFFD1200–0xFFFD12FF	SEC Channel 2 Primary Addresses (used if MCR[RCA2] = 0)	
– 0xFFFD1200–0xFFFD1207	reserved	
– 0xFFFD1208	Channel 2 Configuration Register	CCR2
– 0xFFFD1210	Channel 2 Status Register	CSR2
– 0xFFFD1218–0xFFFD123F	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD1240	Channel 2 Current Descriptor Pointer Register	CDPR2
– 0xFFFD1248	Channel 2 Fetch FIFO	CFF2
– 0xFFFD1250–0xFFFD127F	reserved	
– 0xFFFD1280–0xFFFD12BF	Channel 2 Descriptor Buffer	DB2
– 0xFFFD12C0–0xFFFD12DF	Channel 2 Gather Link Tables	—
– 0xFFFD12E0–0xFFFD12FF	Channel 2 Scatter Link Tables	—
– 0xFFFD1300–0xFFFD13FF	SEC Channel 3 Primary Addresses (used if MCR[RCA3] = 0)	
– 0xFFFD1300–0xFFFD1307	reserved	
– 0xFFFD1308	Channel 3 Configuration Register	CCR3
– 0xFFFD1310	Channel 3 Status Register	CSR3
– 0xFFFD1318–0xFFFD133F	reserved	
– 0xFFFD1340	Channel 3 Current Descriptor Pointer Register	CDPR3
– 0xFFFD1348	Channel 3 Fetch FIFO	CFF3
– 0xFFFD1350–0xFFFD137F	reserved	
– 0xFFFD1380–0xFFFD13BF	Channel 3 Descriptor Buffer	DB3
– 0xFFFD13C0–0xFFFD13DF	Channel 3 Gather Link Tables	—
– 0xFFFD13E0–0xFFFD13FF	Channel 3 Scatter Link Tables	—
– 0xFFFD1400–0xFFFD14FF	SEC Channel 4 Primary Addresses (used if MCR[RCA4] = 0)	
– 0xFFFD1400–0xFFFD1407	reserved	
– 0xFFFD1408	Channel 4 Configuration Register	CCR4
– 0xFFFD1410	Channel 4 Status Register	CSR4
– 0xFFFD1418–0xFFFD143F	reserved	
– 0xFFFD1440	Channel 4 Current Descriptor Pointer Register	CDPR4
– 0xFFFD1448	Channel 4 Fetch FIFO	CFF4
– 0xFFFD1450–0xFFFD147F	reserved	
– 0xFFFD1480–0xFFFD14BF	Channel 4 Descriptor Buffer	DB4
– 0xFFFD14C0–0xFFFD14DF	Channel 4 Gather Link Tables	—
– 0xFFFD14E0–0xFFFD14FF	Channel 4 Scatter Link Tables	—

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD1500–0xFFFD16FF	Polychannel	
– 0xFFFD1500	Fetch FIFO Enqueue Count	FFEC
– 0xFFFD1508	Descriptor Finished Count	DFC
– 0xFFFD1510	Data Bytes In Count	DBIC
– 0xFFFD1518	Data Bytes Out Count	DBOC
– 0xFFFD1520–0xFFFD16FF	reserved	
– 0xFFFD1700–0xFFFD1BF7	reserved	
– 0xFFFD1BF8	Controller IP Block Revision Register	CIPBRR
– 0xFFFD1C00–0xFFFD1FFF	reserved	
– 0xFFFD2000–0xFFFD2FFF	DEU	
– 0xFFFD2000	DEU Mode Register	DEUMR
– 0xFFFD2008	DEU Key Size Register	DEUKSR
– 0xFFFD2010	DEU Data Size Register	DEUDSR
– 0xFFFD2018	DEU Reset Control Register	DEURCR
– 0xFFFD2020–0xFFFD2027	reserved	
– 0xFFFD2028	DEU Status Register	DEUSR
– 0xFFFD2030	DEU Interrupt Status Register	DEUISR
– 0xFFFD2038	DEU Interrupt Mask Register	DEUIMR
– 0xFFFD2040–0xFFFD204F	reserved	
– 0xFFFD2050	DEU End_of_Message Register	DEUEOMR
– 0xFFFD2058–0xFFFD20FF	reserved	
– 0xFFFD2100	DEU IV Register	DEUIVR
– 0xFFFD2108–0xFFFD23FF	reserved	
– 0xFFFD2400	DEU Key 1 Register	DEUKR1
– 0xFFFD2408	DEU Key 2 Register	DEUKR2
– 0xFFFD2410	DEU Key 3 Register	DEUKR3
– 0xFFFD2418–0xFFFD27FF	reserved	
– 0xFFFD2800–0xFFFD2FFF	DEU Input FIFO/Output FIFO	—
– 0xFFFD3000–0xFFFD3FFF	reserved	
– 0xFFFD4000–0xFFFD4FFF	AESU	
– 0xFFFD4000	AESU Mode Register	AESUMR
– 0xFFFD4008	AESU Key Size Register	AESUKSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD4010	AESU Data Size Register	AESUDSR
– 0xFFFD4018	AESU Reset Control Register	AESURCR
– 0xFFFD4020–0xFFFD4027	reserved	
– 0xFFFD4028	AESU Status Register	AESUSR
– 0xFFFD4030	AESU Interrupt Status Register	AESUISR
– 0xFFFD4038	AESU Interrupt Mask Register	AESUIMR
– 0xFFFD4040	AESU ICV Size Register	AESUICVSR
– 0xFFFD4048–0xFFFD404F	reserved	
– 0xFFFD4050	AESU End_of_Message Register	AESUEOMR
– 0xFFFD4058–0xFFFD40FF	reserved	
– 0xFFFD4100	AESU Context Register 1	AESUCR1
– 0xFFFD4108	AESU Context Register 2	AESUCR2
– 0xFFFD4110	AESU Context Register 3	AESUCR3
– 0xFFFD4118	AESU Context Register 4	AESUCR4
– 0xFFFD4120	AESU Context Register 5	AESUCR5
– 0xFFFD4128	AESU Context Register 6	AESUCR6
– 0xFFFD4130	AESU Context Register 7	AESUCR7
– 0xFFFD4138	AESU Context Register 8	AESUCR8
– 0xFFFD4140	AESU Context Register 9	AESUCR9
– 0xFFFD4148	AESU Context Register 10	AESUCR10
– 0xFFFD4150	AESU Context Register 11	AESUCR11
– 0xFFFD4158	AESU Context Register 12	AESUCR12
– 0xFFFD4160–0xFFFD43FF	reserved	
– 0xFFFD4400	AESU Key Upper Register 1	AESUKUR1
– 0xFFFD4408	AESU Key Lower Register 1	AESUKLR1
– 0xFFFD4410	AESU Key Upper Register 2	AESUKUR2
– 0xFFFD4418	AESU Key Lower Register 2	AESUKLR2
– 0xFFFD4420–0xFFFD4FFF	reserved	
– 0xFFFD5000–0xFFFD5FFF	reserved	
– 0xFFFD6000–0xFFFD6FFF	MDEU	
– 0xFFFD6000	MDEU Mode Register	MDEUMR
– 0xFFFD6008	MDEU Key Size Register	MDEUKSR
– 0xFFFD6010	MDEU Data Size Register	MDEUDSR
– 0xFFFD6018	MDEU Reset Control Register	MDEURCR
– 0xFFFD6020–0xFFFD6027	reserved	
– 0xFFFD6028	MDEU Status Register	MDEUSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD6030	MDEU Interrupt Status Register	MDEUISR
– 0xFFFD6038	MDEU Interrupt Mask Register	MDEUIMR
– 0xFFFD6040	MDEU ICV Size Register	MDEUICVSR
– 0xFFFD6048–0xFFFD604F	reserved	
– 0xFFFD6050	MDEU End_of_Message Register	MDEUEOMR
– 0xFFFD6058–0xFFFD60FF	reserved	
– 0xFFFD6100–0xFFFD6140	MDEU Context Registers	MDEUCR
– 0xFFFD6148–0xFFFD63FF	reserved	
– 0xFFFD6400–0xFFFD647F	MDEU Key Registers	MDEUKR
– 0xFFFD6480–0xFFFD67FF	reserved	
– 0xFFFD6800–0xFFFD6FFF	MDEU Input FIFO	—
– 0xFFFD7000–0xFFFD7FFF	reserved	
– 0xFFFD8000–0xFFFD8FFF	AFEU	
– 0xFFFD8000	AFEU Mode Register	AFEUMR
– 0xFFFD8008	AFEU Key Size Register	AFEUKSR
– 0xFFFD8010	AFEU Context/Data Size Register	AFEUCDSR
– 0xFFFD8018	AFEU Reset Control Register	AFEURCR
– 0xFFFD8020–0xFFFD8027	reserved	
– 0xFFFD8028	AFEU Status Register	AFEUSR
– 0xFFFD8030	AFEU Interrupt Status Register	AFEUISR
– 0xFFFD8038	AFEU Interrupt Mask Register	AFEUIMR
– 0xFFFD8040–0xFFFD804F	reserved	
– 0xFFFD8050	AFEU End_of_Message Register	AFEUEOMR
– 0xFFFD8058–0xFFFD80FF	reserved	
– 0xFFFD8100–0xFFFD81FF	AFEU Context Memory	—
– 0xFFFD8200	AFEU Context Memory Pointer Register	AFEUCMPR
– 0xFFFD8208–0xFFFD83FF	reserved	
– 0xFFFD8400	AFEU Key Register 1	AFEUKR1
– 0xFFFD8408	AFEU Key Register 2	AFEUKR2
– 0xFFFD8410–0xFFFD87FF	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFD8800–0xFFFD8FFF	AFEU Input FIFO/Output FIFO **Note:** 0xFFFD8E00 is a special address used for the first write when context is written through the input FIFO.	—
– 0xFFFD9000–0xFFFD9FFF	reserved	
– 0xFFFDA000–0xFFFDAFFF	RNGU	
– 0xFFFDA000	RNGU Mode Register	RNGMR
– 0xFFFDA008–0xFFFDA00F	reserved	
– 0xFFFDA010	RNGU Data Size Register	RNGDSR
– 0xFFFDA018	RNGU Reset Control Register	RNGRCR
– 0xFFFDA020–0xFFFDA027	reserved	
– 0xFFFDA028	RNGU Status Register	RNGSR
– 0xFFFDA030	RNGU Interrupt Status Register	RNGISR
– 0xFFFDA038	RNGU Interrupt Mask Register	RNGIMR
– 0xFFFDA040–0xFFFDA04F	reserved	
– 0xFFFDA050	RNGU End_of_Message Register	RNGEOMR
– 0xFFFDA058–0xFFFDA3FF	reserved	
– 0xFFFDA400	RNGU Entropy Register 0	RNGER0
– 0xFFFDA408	RNGU Entropy Register 1	RNGER1
– 0xFFFDA410	RNGU Entropy Register 2	RNGER2
– 0xFFFDA418	RNGU Entropy Register 3	RNGER3
– 0xFFFDA420	RNGU Entropy Register 4	RNGER4
– 0xFFFDA428	RNGU Entropy Register 5	RNGER5
– 0xFFFDA430	RNGU Entropy Register 6	RNGER6
– 0xFFFDA438	RNGU Entropy Register 7	RNGER7
– 0xFFFDA440–0xFFFDA7FF	reserved	
– 0xFFFDA800–0xFFFDAFFF	RNGU Output FIFO	—
– 0xFFFDB000–0xFFFDBFFF	reserved	
– 0xFFFDC000–0xFFFDCFFF	PKEU	
– 0xFFFDC000	PKEU Mode Register	PKEUMR
– 0xFFFDC008	PKEU Key Size Register	PKEUKSR
– 0xFFFDC010	PKEU Data Size Register	PKEUDSR
– 0xFFFDC018	PKEU Reset Control Register	PKEURCR
– 0xFFFDC020–0xFFFDC027	reserved	
– 0xFFFDC028	PKEU Status Register	PKEUSR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFDC030	PKEU Interrupt Status Register	PKEUISR
– 0xFFFDC038	PKEU Interrupt Mask Register	PKEUIMR
– 0xFFFDC040	PKEU AB Size Register	PKEUIABSR
– 0xFFFDC048–0xFFFDC04F	reserved	
– 0xFFFDC050	PKEU End_of_Message Register	PKEUEOMR
– 0xFFFDC058–0xFFFDC1FF	reserved	
– 0xFFFDC200–0xFFFDC27F	PKEU Parameter Memory A0	—
– 0xFFFDC280–0xFFFDC2FF	PKEU Parameter Memory A1	—
– 0xFFFDC300–0xFFFDC37F	PKEU Parameter Memory A2	—
– 0xFFFDC380–0xFFFDC3FF	PKEU Parameter Memory A3	—
– 0xFFFDC400–0xFFFDC47F	PKEU Parameter Memory B0	—
– 0xFFFDC480–0xFFFDC4FF	PKEU Parameter Memory B1	—
– 0xFFFDC500–0xFFFDC57F	PKEU Parameter Memory B2	—
– 0xFFFDC580–0xFFFDC5FF	PKEU Parameter Memory B3	—
– 0xFFFDC600–0xFFFDC7FF	reserved	
– 0xFFFDC800–0xFFFDC9FF	PKEU Parameter Memory N	—
– 0xFFFDCA00–0xFFFDCBFF	PKEU Parameter Memory E	—
– 0xFFFDCC00–0xFFFDCFFF	reserved	
– 0xFFFDD000–0xFFFDDFFF	STEU	
– 0xFFFDD000	STEU Mode Register	STEUMR
– 0xFFFDD008	STEU Key Size Register	STEUKSR
– 0xFFFDD010	STEU Data Size Register	STEUDSR
– 0xFFFDD018	STEU Reset Control Register	STEURCR
– 0xFFFDD020–0xFFFDD027	reserved	
– 0xFFFDD028	STEU Status Register	STEUSR
– 0xFFFDD030	STEU Interrupt Status Register	STEUISR
– 0xFFFDD038	STEU Interrupt Mask Register	STEUIMR
– 0xFFFDD040–0xFFFDD047	reserved	
– 0xFFFDD048	STEU Data Out Register	STEUDOR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFDD050	STEU End_of_Message Register	STEUEOMR
– 0xFFFDD058–0xFFFDD0FF	reserved	
– 0xFFFDD100	STEU IV1 Register	STEUIV1R
– 0xFFFDD108	STEU ICV_IN Register	STEUICVIR
– 0xFFFDD110	STEU IV2 Register	STEUIV2R
– 0xFFFDD118	STEU Context Register 1	STEUCR1
– 0xFFFDD120	STEU Context Register 2	STEUCR2
– 0xFFFDD128	STEU Context Register 3	STEUCR3
– 0xFFFDD130	STEU Context Register 4	STEUCR4
– 0xFFFDD138	STEU LFSR State Register 0	STEULFSRSR0
– 0xFFFDD140	STEU LFSR State Register 1	STEULFSRSR1
– 0xFFFDD148	STEU LFSR State Register 2	STEULFSRSR2
– 0xFFFDD150	STEU LFSR State Register 3	STEULFSRSR3
– 0xFFFDD158	STEU LFSR State Register 4	STEULFSRSR4
– 0xFFFDD160	STEU LFSR State Register 5	STEULFSRSR5
– 0xFFFDD168	STEU LFSR State Register 6	STEULFSRSR6
– 0xFFFDD170	STEU LFSR State Register 7	STEULFSRSR7
– 0xFFFDD178	STEU FSM State Register 1	STEUFSMSR1
– 0xFFFDD180	STEU FSM State Register 2	STEUFSMSR2
– 0xFFFDD188–0xFFFDD3FF	reserved	
– 0xFFFDD400	STEU Key Data Register 1	STEUKDR1
– 0xFFFDD408	STEU Key Data Register 2	STEUKDR2
– 0xFFFDD410–0xFFFDD7FF	reserved	
– 0xFFFDD800–0xFFFDDFFF	STEU Input FIFO/Output FIFO	—
– 0xFFFDE000–0xFFFDEFFF	KEU	
– 0xFFFDE000	KEU Mode Register	KEUMR
– 0xFFFDE008	KEU Key Size Register	KEUKSR
– 0xFFFDE010	KEU Data Size Register	KEUDSR
– 0xFFFDE018	KEU Reset Control Register	KEURCR
– 0xFFFDE020–0xFFFDE027	reserved	
– 0xFFFDE028	KEU Status Register	KEUSR
– 0xFFFDE030	KEU Interrupt Status Register	KEUISR
– 0xFFFDE038	KEU Interrupt Mask Register	KEUIMR
– 0xFFFDE040–0xFFFDE047	reserved	
– 0xFFFDE048	KEU Data Out Register	KEUDOR
– 0xFFFDE050	KEU End_of_Message Register	KEUEOMR

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFDE058–0xFFFDE0FF	reserved	
– 0xFFFDE100	KEU IV1 Register	KEUIV1R
– 0xFFFDE108	KEU ICV_IN Register	KEUICVIR
– 0xFFFDE110	KEU IV2 Register	KEUIV2R
– 0xFFFDE118	KEU Context Register 1	KEUCR1
– 0xFFFDE120	KEU Context Register 2	KEUCR2
– 0xFFFDE128	KEU Context Register 3	KEUCR3
– 0xFFFDE130	KEU Context Register 4	KEUCR4
– 0xFFFDE138	KEU Context Register 5	KEUCR5
– 0xFFFDE140	KEU Context Register 6	KEUCR6
– 0xFFFDE148–0xFFFDE3FF	reserved	
– 0xFFFDE400	KEU Key Data Register 1	KEUKDR1
– 0xFFFDE408	KEU Key Data Register 2	KEUKDR2
– 0xFFFDE410	KEU Key Data Register 3	KEUKDR3
– 0xFFFDE418	KEU Key Data Register 4	KEUKDR4
– 0xFFFDE420–0xFFFDE7FF	reserved	
– 0xFFFDE800–0xFFFDEFFF	KEU Input FIFO/Output FIFO	—
– 0xFFFDF000–0xFFFDFFFF	CRCU (registers start on **page 27-292**)	
– 0xFFFDF000	CRCU Mode Register	CRCUMR
– 0xFFFDF008	CRCU Key Size Register	CRCUKSR
– 0xFFFDF010	CRCU Data Size Register	CRCUDSR
– 0xFFFDF018	CRCU Reset Control Register	CRCURCR
– 0xFFFDF020	CRCU Control Register	CRCUCR
– 0xFFFDF028	CRCU Status Register	CRCUSR
– 0xFFFDF030	CRCU Interrupt/Error Status Register	CRCUISR
– 0xFFFDF038	CRCU Interrupt/Error Mask Register	CRCUIMR
– 0xFFFDF040	CRCU ICV Size Register	CRCUICVSR
– 0xFFFDF048–0xFFFDF04F	reserved	
– 0xFFFDF050	CRCU End_of_Message Register	CRCUEOMR
– 0xFFFDF058–0xFFFDF0FF	reserved	
– 0xFFFDF100	CRCU Context Register	CRCUCXR
– 0xFFFDF108–0xFFFDF3FF	reserved	
– 0xFFFDF400	CRCU Key Register	CRCUKR
– 0xFFFDF408–0xFFFDF7FF	reserved	

Table 9-8. Detailed System Memory Map (Continued)

Address	Name/Status	Acronym
– 0xFFFDF800– 0xFFFDFFFF	CRCU Input FIFO	—
• 0xFFFE0000– 0xFFFFEFFF	reserved	
0xFFFFF000– 0xFFFFFFFF	reserved	

MSC8156E SC3850 DSP Subsystem 10

Each SC3850 core is embedded in an DSP subsystem that enhances the power of the SC3850 core and provides a simple interface to each SC3850 core. The DSP core subsystem includes:

- SC3850 core.
- Instruction channel with a 32-Kbyte instruction cache that supports advanced prefetching.
- Data channel built around a 32-Kbyte data cache, which supports advanced prefetching.
- Memory management unit (MMU) for task protection and address translation.
- Unified 512-Kbyte L2 cache with partitioning support for multitasking reconfigurable in 64-Kbyte partitions as M2 memory.
- Write queue that interfaces between the core and the data channel
- Dual timer for internal use (such as RTOS).
- Extended programmable interrupt controller (EPIC) supporting 256 interrupts.
- Real-time debug support with the OCE and a debug and profiling unit (DPU).

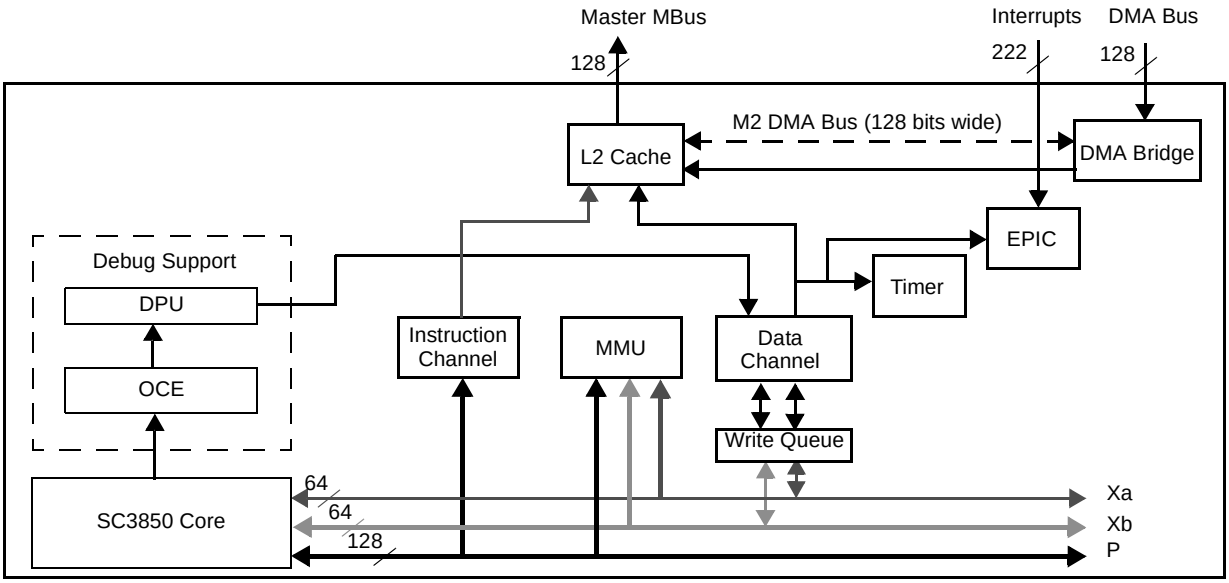


Figure 10-1. DSP Core Subsystem

Note: *The SC3850 DSP Core Reference Manual* and the *MSC8156 SC3850 DSP Core Subsystem Reference Manual* have detailed information on the DSP core and core subsystem. Both manuals are available with a signed non-disclosure agreement. Contact your local Freescale dealer or sales representative for more information.

The remainder of this chapter describes each of these DSP core subsystem components.

10.1 SC3850 DSP Core Subsystem Features

The subsystem has the following units and distinctive features:

- SC3850 DSP core (see **Chapter 2**, *SC3850 Core Overview* for details).
- Instruction cache (ICache):
  - 32 Kbytes
  - 8 ways with 16 lines per way
  - Multi-task support
  - Real-time support through locking flexible boundaries
  - Prefetch capability
  - Software coherency support (‘Sweep’)
  - PFETCH touch loading instruction support
- Data cache (DCache):
  - 32 Kbytes
  - 8 ways with 16 lines per way
  - Can serve two data accesses in parallel (XA, XB)
  - Multi-task support
  - Real-time support through locking flexible boundaries
  - Software coherency support (Cache ISA or “Sweep”)
  - Write-back writing policy
  - Write-through writing policy
  - Hardware line prefetch capability
  - Cache performance ISA support (DFETCH touch loading and DMALLOC)
- Memory management unit (MMU):
  - Virtual to physical address translation
  - Task protection
  - Multi-tasking
  - Defines the memory and access attributes of memory regions
- Unified L2 cache:
  - 512 KB
  - 8 ways with 1024 indices
  - 64-byte line size
  - Physically addressed

- Maximum user flexibility for real-time support through address partitioning of the cache
- Rich cache policy support
- Multi channel, two dimensional software prefetch support
- Software coherency support with seamless transition from L1 cache coherency operation.
- External memory Interface:
- MBus unified address separate data bus, with 32-bit address and 128-bit data
- Supports asynchronous clock ratio

- Debug and profiling:
  - On-chip emulator (OCE) for core-related debug and profiling support
  - Debug and profiling unit (DPU) for subsystem level debug and profiling support
  - Debug state, single stepping and command insertion from the host debugger
  - Breakpoints on PC, data address, and data bus values
  - More than 40 event counting options in 6 parallel counters
  - Cache debug mode enabling to observe the cache state (cache array, tags, valid and dirty bits, and so on) and to change the contents of the data cache array.
  - Real-time tracing of PC, task ID, and profiling information to the main memory
- Interrupt handling:
  - Extended programmable interrupt controller (EPIC) to handle 256 interrupts, including from internal sources
  - Supports 222 interrupts external to the MSC8156 SC3850 DSP subsystem, independently configured as maskable or non-maskable
  - 32 priority levels for interrupts
  - Asynchronous and synchronous interrupts.
- Timer. Two general-purpose 32-bit timers for RTOS support
- Low-power design modes of operation:
  - Wait processing state, where the clocks of the core and caches are gated but peripherals operate.
  - Stop processing state for full clock gating

10.2 SC3850 Core

The SC3850 core is an improved superset of the SC3400 architecture that is binary compatible with the previous StarCore architectures, including the SC140/SC140e and SC3400. The SC3850 core is organized the same way as the StarCore architectures. See **Chapter 2**, *SC3850 Core Overview* and the *SC3850 DSP Core Reference Manual* for details on the SC3850 core.

10.3 Instruction Channel

The instruction channel, which comprises the instruction cache (ICache) and the instruction fetch unit (IFU), provides the core with instructions that are stored in higher-level memory. The ICache operates at core speed and stores recently accessed instructions. Whenever an addressed instruction (from the cacheable memory area) is found in the array, it is immediately made available to the core (ICache hit). When the required address is not found in the array, it is loaded to the ICache from the external (off-subsystem) memory by the IFU (ICache miss). The IFU operates in parallel with the core to implement a HW line prefetching algorithm that loads the ICache with information that has a high probability of being needed soon. This action reduces the number of cache misses. When an instruction is addressed from a non-cacheable area, the IFU fetches it directly to the XP bus of the core without writing it to the cache.

10.3.1 Instruction Cache

The ICache has the following parameters:

- A size of 32 Kbytes with 256 bytes per cache line (total of 128 lines in the cache) arranged in an 8 way set-associative structure.
- A cache line logically divided into 16 valid bit resolutions (VBRs), each 16 bytes long. This is the resolution for hit or miss detection.

Upon a cache miss, the ICache fetches the required information. A pseudo-LRU replacement algorithm is used to select the line to be replaced when required. The flexible ICache has a locking mechanism that can lock some ways, thus partitioning the cache between tasks. The programmer can flush the full contents of the ICache (meaning, invalidate all tags and VBRs) or selectively flush its contents between programmable address boundaries.

The cache fetches a core instruction for touch loading (PFETCH) by queuing it and initiating it when there is a free slot on the program bus. Using this touch loaded instruction can dramatically reduce the average cache miss penalty. In addition, a programmable HW next line prefetch is used to further reduce the miss ratio for code that is sequential in nature.

The ICache supports real-time and non-real-time debugging and profiling. In real time, it provides information on misses and hits. In non-real-time, it supports access to all its internal information, namely, the tag, the replacement status, and the valid arrays. The cache array itself can be either read or written. This information is accessed through the JTAG interface in Debug processing state.

10.3.2 Instruction Fetch Unit

The IFU controls the fetching of instructions from cacheable external memory when a miss indication is received from the ICache. It controls information updates in the ICache. The programmer can control the IFU HW line prefetching to adjust it for better performance for an application. For a request from a non-cacheable area, the information from the external memory is made available directly to the core through the XP bus. To minimize the idle time of the core, the IFU implements critical word first external access and also supports prefetch hit.

10.4 Data Channel

The data channel comprises the data cache (DCache), the data fetch unit (DFU), the data control unit (DCU), the write-back buffer (WBB), and the write-through buffer (WTB). This two-way channel reads and writes information from the core to/from higher-level memory (M2 or L2) and control memory (internal blocks and external peripherals) spaces.

The DCache, which operates at core speed, keeps the recently accessed data. When addressed data (from a cacheable memory area) is found in the array, it is immediately made available to the core (DCache hit) in a read and updated if written to. When the required address is not found in the array, a DCache miss occurs, and the DFU loads the data to the DCache from the external (off-subsystem) memory and drives it to the core. The DFU operates in parallel with the core and implements a HW line prefetch algorithm that loads the DCache with information that has a high probability of being needed soon, thus reducing the number of data cache misses.

The channel differentiates between cacheable and non-cacheable addresses. For cacheable addresses, it supports the write-back allocate and write-through writing policies. The selection is made on an address segment basis, as programmed in the MMU. The data channel supports the arrangement of data in big-endian formats. Core data types can be byte, word, long (4 bytes), or 2 long (8 bytes) wide.

10.4.1 Data Cache

The DCache has the following parameters:

- A size of 32 Kbytes with 256 bytes per cache line (total of 128 lines in the cache) arranged in an 8 way set-associative structure.
- A cache line logically divided into 16 VBRs, each 16 bytes long. This is the resolution for hit or miss detection. Big-endian byte arrangements are supported.

A data access that is identified as a hit in the DCache is served without freezing the core, with the exception of DCache contention (dual access from two core buses to the same module). If the requested data is not in the cache, the DFU fetches it. If the requested data belongs to a cache line that is not in the cache, the line is allocated at the expense of another line. A pseudo-LRU (PLRU) replacement algorithm selects the line to be replaced. If the line to be replaced contains

valid and dirty VBRs, it must be written (thrashed) to the external memory. The DCache implements the write-back and write-through writing policies. To speed up the freeing of the cache line, a write is directed to the write-back buffer and later copied from there to the external memory.

The DCache enables the system/software architect to control its operation. A locking mechanism with global locking or partial locking helps reduce the penalty of restoring critical tasks.

A great deal of control flexibility is offered by the sweep command, by which information in the DCache is managed according to its virtual address. The programmer defines a virtual address range and a specific operation to be carried out on cache lines within that range. The operation can involve line invalidation, line synchronization, or line flush. The operation can execute in parallel with core operation. The core can be interrupted at the end of the operation. In addition to sweep, the cache supports SC3850 coherency instructions Flush memory block (DFLUSH) that writes back and invalidates a block of 64 bytes belonging to the specified address and Synchronize memory block (DSYNC) that writes back a block of 64 bytes belonging to the specified address back to memory.

The cache supports touch loading data fetch core instructions (DFETCH) by queuing them and initiating them when there is a free slot on the data bus. Using this instruction can dramatically reduce the average cache miss penalty.

The cache also supports the data memory allocation instruction (DMALLOC), which causes the cache to allocate a line matching the designated address and validating a 64 byte resolution (wrapped on 64-byte boundary) with almost zero overhead without actually fetching the data from memory.

The DCache supports real-time and non-real-time debugging and profiling. In real time, it provides information on misses, hits, and other cache-related events. In non-real time (debug processing state), it enables access to all of its internal information, such as the tag, the PLRU, the valid arrays, and the DCache array. The cache array can be written to as well. This information is accessed through the JTAG interface in debug processing state.

10.4.2 Data Fetch Unit

The DFU controls data fetches from cacheable external memory when it receives a miss indication from the DCache. The unit controls information updates in the DCache. The programmer can control its HW line prefetch to adjust for better performance for an application. For a request from a non-cacheable area, the information from the external memory is made available directly to the core through the requesting bus, WA or WB without updating the cache. The programmer can control the burst size, as well as turn the HW line prefetch mechanism on or off. To minimize the time that the core stalls, the DFU implements critical word first external access and also supports prefetch hit.

10.4.3 Write-Back Buffer

The WBB expedites freeing of the DCache to enable fetching of new data with minimal core delay. Modified data from the DCache array is transferred to the WBB, thus freeing the array for additional data fetch from a higher-level memory. This effectively splits the thrashing process into two separate operations. During the first operation, the data to be thrashed is stored in the WBB until after the fetch operation completes. The second operation writes the data to be thrashed out to the higher-level memory.

The WBB has the size of 16 VBRs arranged as 8 entries of 2 VBRs each. It operates as a FIFO buffer. It sends data to the external memory using the QBus protocol through the DCU. It provides the DCU with priority information, either normal or high. Normal priority is used for standard DCache line replacement. High priority is used for the DCache flushing operation.

The WBB also snoops the external core accesses (the physical addresses that are translated from the virtual addresses by the MMU) to detect a hazard situation in which a request is made for a line in the WBB before it gets to the higher-level memory. When such a situation occurs, the WBB is flushed as a high priority request for updating the higher-level memory before fetching from the memory continues. This ensures data coherency.

10.4.4 Write-Through Buffer

The WTB provides a short route for writing data to the data QBus memory space, meaning to external (off-subsystem) memory through the MBus and to internal subsystem registers and external peripherals. The WTB block generates write accesses from the write queue output buses to the devices connected to the data QBus when the application requires a non-cacheable or write-through write access. The WTB also serves write accesses when the cache is disabled, in debug, or missed when in lock. The WTB is arranged as six 64-bit entries. It operates as a FIFO buffer to buffer the latency of the write accesses passing through it from the DSP core.

10.4.5 Data Control Unit

The DCU prioritizes and arbitrates between the various write transactions (from the WBB, the WTB, and the trace writes from the TWB) and the read transactions of the DFU. The DCU transfers the transactions to the data QBus after mastership on the bus is obtained or directly when it accesses the internal subsystem memory-mapped registers.

10.4.6 Write Queue

The write queue (WRQ) interfaces between the SC3850 core and the data channel. The WRQ stores up to 14 write accesses from both the XA and the XB buses and has the following primary functions:

- Bridge the core pipeline gap between the address generation and the data drive.
- Enable read accesses to bypass write accesses (unless there is a read after write hazard).
- Store write accesses (already with data) until they have an available memory slot, thus freeing the core to execute instructions.
- Identify a read-after-write hazard and forward newer write data of pending accessed to replace the older data bytes were read from memory.
- Handles core DCache instructions like DFETCH, DSYNC, DMALLOC, and others.
- Buffer DFETCH and write-back misses that the system currently can not handle without stalling the core until the WRQ is full.

10.5 Memory Management Unit (MMU)

The MMU performs three main functions:

- Memory hardware protection for instruction and data access with two privilege levels (user and supervisor).
- High-speed address translation from virtual to physical address to support memory relocation.
- Cache and bus controls for advanced memory management

Memory protection increases the reliability of the system so that errant tasks cannot ruin the privileged state and the state of other tasks. Program and data accesses from the core can occur at either the user or supervisor level. The MMU checks each access to determine whether it matches the permissions defined for this task in the memory attributes and translation table (MATT). If it does not, the access is killed and a memory exception is generated.

The MMU performs address translation on external (off-subsystem) addresses, from virtual addresses (used by the software that runs on the core) to physical addresses (used by the system buses). Benefits of address translation include the following:

- Enables software to be written without consideration of the physical location of the code in memory, thereby providing a simpler software model that enhances modularity and reuse.
- Allows true dynamic code relocation without performance cost or overhead.

The same virtual addresses can be reused between tasks without a need to flush the caches between tasks because the caches store the task ID in their line tags and thus have a unique

memory image per task. Protection and address translation are applied to memory segments defined in the MMU. A segment descriptor (SD) can set cacheable/non-cacheable, prefetch policy, shared/non-shared, and more. The MMU controls up to 20 data and up to 12 program segment descriptors.

The MSC8156 SC3850 DSP subsystem introduces a new programming model for more flexible memory mapping of the SD that reduces previous constraints allowing reduction in the number of descriptors needed and memory waste. For compatibility, the MMU also supports the old programming model.

10.6 L2 Cache

The L2 cache processes data and program accesses to the external M3/DDR memory. Caching the accesses requested by the L1 subsystem reduces the average penalty of accessing the high latency M3. The L2 cache includes a slave arbitration and tag unit, cache logic and arrays, along with a write buffer for write back and write through accesses, fetch logic to fetch data from the off platform memory upon a miss or a non-cacheable access, and a master arbiter that arbitrates between the different internal units.

Features of the L2 cache are as follows:

- A size of 512 Kbytes with 64 bytes per cache line (total of 8192 lines in the cache) arranged in an 8 way set-associative structure.
- 64Byte valid bit resolution (VBR) and dirty bit resolution (DBR). Fetch the whole line at once. Write back the whole line at once when a dirty line is thrashed from the cache.
- Physically addressed
- Dynamically configurable as a DMA accessible M2 RAM
- Software coherency support with seamless transition from L1 cache coherency operation
- Multichannel (4) L2 software prefetch for two-dimensional arrays
- Cache partitioning by way
- Write policies:
  - Non-cacheable and non-cacheable on read and write (NC)
  - Cacheable write-through and cacheable on read non-cacheable on write (WT). Update the cache on hit.
  - Cacheable write back; both read and write are cacheable (WB)
  - Adaptive write policy (AWP). Cacheable WB on hit and NC on miss.
- WB composed of eight 32-byte entries
- Full ECC support

The main components of the L2 cache are as follows:

- L2 Qbus arbiter (L2Q) to arbitrate the input data QBus, instruction QBus, and the slave port to one output bus (necessary because the cache serves one access per cycle).
- Cache logic to manage the cache arrays and state machines
- Fetch unit (L2FU) to fetch cacheable accesses from the M3/DDR memory, including L2 SW prefetching of data and instructions not yet requested by the core.
- Write buffer (L2WB) to temporarily hold modified (dirty) cache lines thrashed by the cache and also serve as a buffer for non-cacheable memory.
- Arbitration unit (L2AU) to arbitrate among the different masters (access generators) of the L2 cache (L2FU and L2WB).
- QBus to MBus interface (Q2M) to interface the external memory IF bus asynchronously while maintaining the pipeline.
- Register file to hold the L2 cache status and control registers. Mapped to the internal peripherals on bank0.

10.7 On-Chip Emulator and Debug and Profiling Unit

The on-chip emulator (OCE) and the debug and profiling unit (DPU) are hardware blocks for debugging and profiling. The OCE performs the following tasks:

- Communicates with the host debugger through the SoC JTAG test access port (TAP) controller
- Enables the SC3850 core to enter the debug processing state upon a varied set of conditions to:
  - Single step
  - Execute core commands inserted from the host debugger to upload and download memory and core registers.
- Sets up to six address-related breakpoints on either PC or a data address
- Sets a data breakpoint on a data value, optionally combined with a data address
- Generates the PC tracing flow, optionally filtered to a subset of events such as only jumps/returns from subroutine, interrupts, and so on.

The DPU has the following characteristics:

- Enables parallel counting of subsystem events in six dedicated counters, from more than 40 events
- Filters, processes, and adds task ID and profiling information on the OCE PC trace information

10.8 Extended Programmable Interrupt Controller

The internal extended programmable interrupt controller (EPIC) manages internal and external interrupts. The EPIC handles up to 256 interrupts, 222 of which are external subsystem inputs. The rest of the interrupts serve internal subsystem conditions. The external interrupts can be configured as either maskable interrupts or non-maskable interrupts (NMIs). The EPIC can handle 33 levels of interrupt priorities, of which 32 levels are maskable at the core and 1 level is NMI.

10.9 Timer

The timer block includes two 32-bit general-purpose counters with pre-loading capability. It counts clocks at the core frequency. It is intended mainly for operating system use.

10.10 Interfaces

The interfaces transfer data from the subsystem to the rest of the system and vice versa.

10.10.1 QBus to MBus Interface Bridge

The instruction QBus to MBus interface (IQ2MA) bridge translates the bus from the L2 Cache, which uses the proprietary QBus protocol, to the MBus line protocol. The Q2M is placed between two different asynchronous clock domains:

- Internal, MSC8156 SC3850 DSP subsystem clock domain
- External (out of subsystem) clock domain, which is slower or equal to the internal clock domain.

10.10.2 MBus to DMA Bridge

The MBus to DMA bridge converts the signals coming from the MBus to the protocol used by the L2 Cache slave port. The bridge is placed between two different asynchronous clock domains:

- Internal, MSC8156 SC3850 DSP subsystem clock domain
- External (out of subsystem) clock domain, which is slower or equal to the internal clock domain.

10.11 Entering and Exiting Wait and Stop States Safely

The following subsections describe how to enter and exit from the Wait and Stop states safely.

10.11.1 Wait State

The Wait state is described in **Section 2.12.4** of the *MSC8156 SC3850 DSP Subsystem Reference Manual*. The subsystem enters the Wait processing state when the SC3850 core executes the **wait** instruction. The subsystem exits from the Wait state when an enabled level interrupt request is asserted or the subsystem transfers to the Debug or Reset state. During a Wait state, the subsystem L2/M2 memory is available for access through the slave port. No further steps are required.

10.11.2 Stop State

The Stop state is described in **Section 2.12.5** of the *MSC8156 SC3850 DSP Subsystem Reference Manual*.

10.11.2.1 Procedure for Entering DSP Subsystem Stop State Safely

Note: Before entering the Stop state, terminate any L2 software prefetch activities to reduce the time needed to enter Stop.

Use the following procedure to enter the Stop state:

1. Stop all accesses to the M2/L2 memory in the core subsystem by peripherals and external hosts (if applicable).
2. Close the subsystem slave port window by writing a 1 to the relevant bit in the CORE_SLV_GCR (see **Section 8.2.34**, *Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR)*, on page 8-58 for details).
3. Read the CORE_SLV_GCR to make sure that the 1 is written and the bit is set.

Note: After the slave port window is closed, all core accesses will generate an error interrupt to the core.

4. Send a read request from the specified core and subsystem to an adjacent core M2 according to its index, as follows: from Core0 to Core1, from Core1 to Core0, from Core2 to Core3, or from Core3 to Core2.
5. Step 4 generates an address error interrupt. Read GIR5 to make sure that the interrupt is generated by the specified core (see **Section 8.2.25**, *General Interrupt Register 5 (GIR5)*, on page 8-39 for details).
6. Make sure that the appropriate Stop ACK is asserted in GSR1. If not, assert the bit. (see **Section 8.2.3**, *General Status Register 1 (GSR1)*, on page 8-5 for details).

7. Issue a **stop** command to the specified core.

10.11.2.2 Procedure for Exiting the Stop State Safely

Use the following process to exit from the Stop state:

1. Initiate an enabled level subsystem interrupt. This causes the subsystem to exit the Stop state.
2. The core will open its own slave port window by deasserting the relevant bit in the CORE_SLV_GCR (see **Section 8.2.34**, *Core Subsystem Slave Port General Configuration Register (CORE_SLV_GCR)*, on page 8-58).

Internal Memory Subsystem 11

The internal memory system supports 4.5 MB of internal memory and includes:

- Memory management unit (MMU) per core.
- Instruction channel with 32 KB L1 ICache per core.
- Data channel with 32 KB L1 DCache per core.
- 512 KB L2 shared unified Cache, configurable in 64 KB blocks as M2 memory, per core.
- 1 MB + 32 KB 128-bit wide M3 memory connected to three internal memory buses. The 1 MB block can be turned off to reduce power consumption. The M3 memory supports semaphores and the RapidIO mail boxes.
- 96 KB of boot ROM accessible from the cores.

Note: The MMU, L1 ICache, L1 DCache, and L2 Cache/M2 memory are part of the MSC8156E SC3850 DSP core subsystem. For detailed programming and functional information, refer to the *MSC8156 SC3850 DSP Core Reference Manual*, available with a signed non-disclosure agreement. Contact your local Freescale dealer or sales representative for more information.

Note: The default burst size for the caches and MMU is 1 VBR, but for most operations, programming the burst size as 4 VBR provides the best performance.

11.1 Memory Management Unit (MMU)

The MMU provides a high-speed address translation mechanism to enable memory relocation, and checks access permissions for core instructions and data buses. It also controls hardware task protection and provides cache and bus controls for advanced memory management. The MMU enables better integration of system resources and defines a cleaner software model. For example, programing protected regions, address translation regions, cacheable regions, and so on can be combined. In addition, cache usage can be optimized based on the specific attributes controlled by the MMU programming. For memory protection, the MMU enables the implementation of an RTOS with MMU support, thereby protecting the operating system, task code, and data from errant tasks. Address translation enables implementation of a software model in which the code uses virtual addresses that are translated to physical addresses accessing memory. The MMU provides a virtual memory software model with a hole for the OCE and internal device registers and peripherals. The core generates virtual addresses during its operation. The virtual address together with the task ID from the MMU become the task-extended (TE) virtual address. The MMU translates between virtual and physical addresses during each core access, providing control attributes for each core access per memory segment, such as burst size, prefetch enable, write-policy, cacheability, and so forth.

The MMU has the following functions and features:

- A memory attributes and translation table (MATT), composed of 20 data segment descriptors and 12 program segment descriptors.
- Each segment descriptor defines a related memory region and its cache and attributes, protection and address translation.
- The descriptor related memory space has a long-range variable mapping size. The size is designated in steps as a power of 2, starting from 256 bytes. The mapping size can be between 256 bytes to 4 GB. The base address must be aligned to a segment size.
- The memory region dedicated cache and attributes support the following:
  - Cacheable access.
  - A burst size of 1, 2, or 4 for the data fetch unit (DFU) and instruction fetch unit (IFU).
  - Hardware line prefetch enable.
  - Hardware next line prefetch enable (instruction only)
  - System/shared attributes
  - Write policy for data memory
  - L2 cache policy

- Hardware data and program access protection is defined for each data/program memory region for two privilege levels: user and supervisor. The MMU provides abort signals for the Xa/Xb/P buses for errant accesses. The MMU provides memory region support, as follows:
  - For the program memory region, provides read allowed/not allowed access for both the Supervisor and User levels
  - For the data memory region, provides read/write allowed/not allowed for both the Supervisor and User levels
- Address translation is defined for each data/program memory region, enabling allocation of a virtual memory region to a valid physical memory space.
- Priority mechanism between descriptors, allowing memory regions overlapping.
- Stores the program task ID and data task ID for multi-task mechanism. Up to 255 different Program IDs and 255 different data IDs are available.
- Subsystem ID register and general-purpose register among its control and status registers.
- Access error detection including non-mapped memory access and misaligned memory access.
- Captures error status bits and enables a fast error diagnostic.
- Precise interrupts allowing handling MMU MATT misses supporting a virtually paged operating system.
- Core branch target buffer (BTB) that enables manual and automatic BTB maintenance.
- Subsystem error detection code (EDC) recovery scheme.
- Enable/disable EDC exception mechanism.
- Subsystem master and slave error handling.
- Program and data query mechanism.

11.2 Instruction Channel (ICache and IFU)

The Instruction Channel comprises the Instruction Cache (ICache) and the Instruction Fetch Unit (IFU). This channel provides the core with instructions that are stored in higher-level memory. The ICache, which operates at core speed, stores recently accessed instructions. Whenever an addressed instruction (from the cacheable memory area) is found in the array, it is immediately made available to the core (ICache hit). When the required address is not found in the array, it is loaded to the ICache from the external (not part of the DSP core subsystem) memory by the IFU (ICache miss). The IFU operates in parallel to the core to implement a prefetch algorithm that loads the ICache with information that with high probability will be needed soon. This action reduces the number of cache misses. Whenever an instruction is addressed from a non-cacheable area, the IFU fetches it directly to the P bus of the core without writing it to the cache.

Instruction channel features include:

- Aligned 128-bit XP core accesses.
- Concurrent cacheable and non-cacheable accesses, as identified in the MMU based on the address ranges.
- Task-extended virtually addressed cache. The 8-bit task ID from the MMU is stored as part of the line tag that allows a task-specific cache image that is not overridden by other tasks that use the same virtual address. This feature can support multi-task mechanism. This extended tag is named the ETAG.
- Cacheable shared memory between tasks marked by the MMU according to the memory range, and stored in the cache with TASKID 0.
- Cache hit access without wait states (except during memory conflicts).
- Issues 128-bit accesses to the higher level memory when a cache miss occurs.
- Programmable to issue hardware line prefetch accesses upon cache miss that fetch data to the end of the cache line (256 bytes).
  - Hardware line prefetching is aborted in case of a new miss on a burst size boundary.
  - Programmable burst size of 1, 2, or 4 VBRs.
- A miss access identified as a prefetch by the prefetch hit stalls the core to reduce the number of wait states relative to a simple miss.
- Automatic hardware next line prefetch on a mis or a hit to a previously fetched line controlled by the MMU.
- Supports SC3850 core touch loading PFETCH command that allows the user to prepare specific memory lines in the cache to reduce or remove the penalty for miss accesses.
- Pseudo-LRU (PLRU) as the cache line replacement mechanism (LRM).
- Partial lock allows locking of a subset of cache lines based on ways boundaries to reduce the cache restoration penalty of a restored task. Instructions can be locked in the cache, preventing the thrashing of instructions that have expected reuse. This mechanism is useful during rapid task switching and prevents a situation in which a task thrashes important instructions associated with other tasks.
- Global lock allows locking of all cache lines to reduce the cache restoration penalty of a restored task. In this case, the cache does not serve cache misses.
- User-initiated cache sweep operations for coherency support. These operations are performed on each line in a user-specified address range. An invalidate discards the cache line (clears the valid bits).
- Cache debug mode in which the cache state (ETAG values, Valid, PLRU state) can be read and the memory array can be read or written.
- Dedicated programmable cache control registers that control or reflect its operation.
- EDC (error detection) support.
- Dedicated exceptions for each of the following events:

— *End of sweep operation*. This exception indicates the completion of the sweep operation.
— *XP non-cacheable hit access*. This exception indicates that an access is a hit access, even though the MMU classifies it as a non-cacheable access. This type of situation can occur if the memory space attributes changed in the MMU without invalidating the appropriate cache lines Assertion of the exception is not guaranteed when the attribute change that led to this error is not performed with SYNCIO as restricted.
— *XP double match.* This is an error that occurs when a task-shared access has an address that matches a non-shared cache line.

11.3 Data Channel and Write Queue (DCache)

The data channel and write queue is a two-way channel for reading and writing information from the core to/from higher level memory (M2 or L2) and Control Memory (internal blocks and external peripherals) spaces. The DCache, which operates at core speed, keeps the recently accessed data. Whenever addressed data (from a cacheable memory area) is found in the array, it is immediately made available to the core (DCache hit) in a read, and updated if written. When the required address is not found in the array, a DCache miss occurs, and the data is loaded to the DCache from the external (not part of the DSP core subsystem) memory by the DFU, and driven through the write queue to the core.

The data channel is fed by accesses from the write queue buses and not directly from the core XA/XB data buses. Data reads are normally forwarded directly from the write queue, so there is no delay in processing them. On the other hand, data writes can be delayed for extended periods in the write queue. Therefore, they are bypassed by read accesses and their processing in the data channel is completely detached from the core execution flow. The write queue performs the necessary hazard checks and enforces the access order issued by the core.

The DFU operates in parallel with the core and implements a prefetch algorithm to load to the DCache, information that with high probability will be needed soon, thus reducing the number of data cache misses. The channel differentiates between cacheable and non-cacheable addresses as defined by the MMU. For cacheable addresses, it supports the write-back allocate or write-through accesses. Cacheable accesses activate the cache (if the cache is enabled) and respond to the core with no wait states when a hit occurs. For a miss, the data channel issues a fetch request to higher-level memory and typically executes hardware line prefetches limited at most by the end of the cache line. Write-through accesses do not allocate a new line in the cache for a write-miss access.

Cacheable write back data writes wait in the cache until a line is flushed, either automatically as part of the replacement policy or when initiated by the user. During flushing, the writes wait in the write back buffer until they are written to higher-level memory. Hardware line prefetching and write backs are issued on the QBus as compact 128-bit transactions, which helps reduce the traffic on the DSP subsystem connection to the higher level memories. Non-cacheable accesses are written through the data channel without changing the cache state. Cacheable write-through accesses, however, are written to the higher level memory, and they update the cache when there is a hit access.

The data channel has the following features:

- Handling 2 parallel core accesses WA/WB each with a width of 1, 2, 4 or 8 bytes
- Supports both cacheable and non-cacheable accesses concurrently, as identified in the MMU based on their address ranges.
- Task-extended virtually addressed cache. The 8-bit task ID from the MMU is stored as part of the line tag, which allows a task-specific cache image that is not overridden by other tasks that use the same virtual address. This feature can support multi-task mechanism. This extended tag is named ETAG.
- Supports cacheable shared memory between tasks that is marked by the MMU according to the memory range, and is stored in the cache with task ID 0.
- Serves a cache hit access without wait states (except memory conflicts).
- Upon a cache miss, issues 128-bit accesses to the higher level memory
- Upon a cache miss, could be programmed to issues prefetch accesses that will bring in data until the end of the cache line (256 bytes).
  - Hardware line prefetching is aborted when there is a new miss on a burst size boundary.
  - Programmable burst size of 1, 2, or 4 VBRs.
- Miss access, identified as being hardware line prefetched (prefetch hit), stalls the core for a reduced number of wait states relative to a simple miss.
- Supports SC3850 core cache maintenance instructions that are operable for both user and supervisor levels:
  - DMALLOC. Allocate a line used later in the code and validate 4 VBRs in the line. Saves time and bus resources need to fetch VBRs from main memory when going to overwrite it.
  - DFETCH/W. Allocate a line and fetch the relevant data (at least one memory burst) that is used later in the code. Adding the W indicate to the L2 cache not to allocate the data even if it is cacheable to reduce cache inclusiveness.
  - DFLUSH. The DCache writes back and invalidates a cache line belonging to the specified address of the M3 or external memory.
  - DSYNC. The DCache writes back a cache line belonging to the specified address of the M3 or external memory.

- Pseudo-LRU (PLRU) as the cache Line Replacement Mechanism (LRM).
- Partial lock allows locking of a subset of cache lines based on ways boundaries, to reduce cache restoration penalty of a restored task. Data can be locked in the cache, thus preventing the thrashing of data expected to be used again. This mechanism is useful when rapid task switching is required, thereby preventing a situation in which a task thrashes important data associated with other tasks.
- Global lock allows locking of all cache lines to reduce cache restoration penalty of a restored task. Miss accesses are not served by the cache in this case.
- Supports user-initiated cache sweep operations for coherency support. These operations are performed on each line in a user-specified address range:
  - Synchronize: write back the cache line if it was modified, clearing its “dirty” bit without affecting its validity.
  - Flush: write back any cache line (and clear the “dirty” bit), and also invalidate it in the cache (clearing the “valid” bit).
  - Invalidate: discard the cache line without writing it back (clear both “dirty” and “valid” bits).
- Cache debug mode where the cache state (ETAG values, Valid-Dirty, PLRU state) could be read and the memory array could be read or written.
- Dedicated programmable cache control registers that control or reflect its operation.
- EDC (error detection) support.
- Provides dedicated exceptions for each of the following events:
  - End of sweep operation: This exception indicates the completion of the sweep operation.
  - WA/WB non-cacheable hit access: This exception indicates that an access is a hit (or prefetch hit) access, although it is indicated by the MMU as a non-cacheable access. This type of situation can occur if the memory space attributes are changed in the MMU without flushing/invalidating the appropriate cache lines Assertion of the exception is not guaranteed if the attribute change that leads to the error was not done with the SYNCIO command as restricted.
  - WA/WB double match: This is an error that occurs when a task-shared access has an address that matches a non-shared cache line.

11.4 L2 Unified Cache/M2 Memory

Each core subsystem contains a memory region that can be allocated as L2 unified cache or M2 memory. Allocation of M2 memory is done in 64 KB blocks from 0 to 512 KB (the full memory space). Whatever portion is not allocated as M2 memory is used as L2 cache. Allocation can be done dynamically, but must be done when the L2 cache is disabled. The selection of 64 KB blocks is done through address partitioning. After the M2 region is defined, the cache should be flushed and after the flush is completed, the L2 cache controller is enabled and the defined M2 region becomes available for addressing with the remainder of the space available to the L2 cache controller (L2Cache).

The L2Cache is responsible for processing data and program accesses to the M3 and external DDR memory. By caching the accesses requested by the L1 subsystem, the average penalty of accessing the high latency M3/DDR memory is reduced. The L2Cache includes slave arbitration and a Tag unit, cache logic and arrays along with a write buffer for write back and write through accesses, fetch logic that is responsible for fetching data from the M3/DDR memory on a miss or a non-cacheable access, and a master arbiter that arbitrates between the different units.

The main L2Cache components include the following:

- Cache logic. Responsible on managing the cache arrays and state machines.
- Fetch Unit (L2FU). Responsible for fetching cacheable accesses from the M3/DDR memory, including the rest of the line.
- Write Buffer (L2WB). A buffer for temporarily holding modified (dirty) cache lines that were thrashed by the cache and also serving as a buffer for non-cacheable memory.
- Control Unit (L2AU). Responsible for arbitration between the different masters (access generators) of the L2Cache (L2FU and L2WB) and interface to the asynchronous bus.
- QBus to MBus (Q2M). Enables the interface to the external memory bus asynchronously while maintaining the pipeline.
- Register file. Holds the L2Cache status and control registers that are mapped to the internal peripherals on Bank0.

The cache has the following characteristics:

- Up to 512 KB cache memory.
- 8 ways
- 1024 cache indexes.
- 64-byte cache line.
- 64-byte Valid Bit Resolution (VBR) and Dirty Bit Resolution (DBR). Fetches the whole line at once. Writes back the whole line at once when a dirty line is thrashed from the cache.
- 8192 cache lines (TAGs).

- Physically addressed.
- L2WB comprises eight 32-byte entries.
- Slave port support.

The L2 Cache Controller is connected to the Data and Instruction QBus. The L2 Cache Controller has a chip select input that signals the access to it. Each access is qualified by the cache policy bits that are linked to the access address, as either a cacheable (CA) access, non-cacheable (NC) access, cacheable write-through (WT), and Adaptive Write Policy (AWP), that is, a CA on a read access or a hit access, and NC on a write miss. Cacheable accesses activate the cache and, in case of a hit, answer after a minimum of five cycles. In case of a miss, the L2 Cache Controller issues fetch requests to the higher-level memory (M3/DDR), and fetches more data than asked for by requesting the whole cache line to reduce the performance impact due to subsequent accesses. In case the access is a WB or AWP hit, the data writes wait in the cache until the line is flushed (automatically as part of the replacement policy or if initiated by the user). Upon flushing, the writes wait in the Write Buffer (L2WB) until they are written to the higher-level memory. Miss fetch and write backs are issued on the MBus as 128-bit transactions, which help reduce the traffic on the platform connection to the higher level memories. NC accesses, WT or AWP miss are written through the L2Cache without changing the cache state via the L2WB. The cache does not calculate hits for NC accesses. To prevent a coherency problem, the cache must be flushed when changing an area from CA policy to NC policy in the MMU MATT descriptors.

The L2 Cache Controller has the following features:

- Input ports (DQBus, IQBus, and MBus):
  - Handling 2 input QBuses and 1 input MBus, arbitrating them to one internal bus with pipeline support.
  - Slave ports (Qbus and MBus) supporting partial bus width accesses of 1, 2, 4, and 8 bytes and full bus bursts of 1, 2, and 4 beats of 128-bit accesses.
  - Round-Robin arbitration according to the access type signal and the origin of the access.
  - Physically addressed.
- Output Port (MBus):
  - Interface to the external memory (Master bus) with a bus working in MBus protocol and supporting asynchronous connection.
  - The maximum accumulative burst size is 64 bytes. The number of beats in the burst is equal to the burst size divided by the bus size.The maximum accumulative burst size is 64 bytes which is made in 4 beats of 128 bits.

- Access handling:
  - Supports the following cache policies:
    - Noncacheable on read and write (NC).
    - Cacheable write-through on read noncacheable on write (WT)
    - Cacheable write back on both read and write are cacheable (WB)
    - Adaptive write policy (AWP). Cacheable WB on read or hit and NC on write miss.
    - Adaptive read cacheability according to a read-with-intent-to-write indiction.
    - Policy is identified by the cache policy bits (defined by MMU programming) that are part of the accesses attributes signals.
  - Supports user-initiated D/PFL2_x commands.
  - Upon a cache miss, brings the entire line using critical word first and wraps on 64-byte boundaries.
  - Identifies data that is being fetched (prefetch hit), which reduces the number of wait states relative to a simple miss and reduces the external memory bus load.
  - Detects hazards for reads that use data that was flushed (or noncacheable) and is still in the L2WB. The access is stalled until the write is completed.
  - Supports fast write and response request
  - Issues a transfer acknowledge as soon as the write data is sampled. When response is high, issues the transfer acknowledge signal when the write access is sampled at the end target.
  - Supports core atomic accesses to external memory. Atomic access to L2 cacheable location or to L2 as M2 is not supported (will always succeed)
  - Constantly memory maps the cache array to enable read and write to it for debug purposes and also to allow use as M2 with DMA connectivity.
- Replacement mechanism:
  - Uses random cache line replacement mechanism (LRM).—Partitioning mechanism by ways and address ranges
  - Allows assignment of a subset of cache lines to an address range to reduce cache restoration penalty of a restored task. This mechanism is useful when rapid task switching is required, thereby preventing a situation in which a task thrashes important data or instructions associated with other tasks.
- ECC is able to fix 1 error in 64 bits. In addition, ECC test mode is supported. This mode enables the user to check the ECC mechanism by initiation of error.
- Cache programming:
  - Dedicated programmable cache control registers that control or reflect its operation.
  - Supports reading and writing memory mapped registers through memory mapped bank0.
- Supports user-initiated SW-PF (L2 software prefetch) operation. This operation enables PF of a specific (two-dimensional) address space as programmed in the cache registers.

- Supports SW cache coherency:
  - SW initiated DFLUSH/DSYNC operations. These operations are performed in a line resolution.
  - SW initiated cache sweep operations for coherency support. These operations are performed on each line in a user-specified address range:
    - Synchronize: write back the cache line if it was modified, clearing its dirty bit without affecting its validity.
    - Flush: write back any dirty cache line (and clear the dirty bit), and also invalidate it in the cache (clearing the valid bit).
    - Invalidate: discard the cache line without writing it back (clear both dirty and valid bits).
- Cache debug mode where the cache state (tag, valid, and dirty) could be read through the DQbus slave port.
- Provides dedicated interrupts for each of the following events:
  - Master port () error: This interrupt indicates that one of the accesses generated by the L2 cache to the off platform memories has failed (that is, non-mapped access).
  - End of sweep operation: This interrupt indicates the completion of the sweep operation.
  - End of SW-PF operation: This interrupt indicates the completion of a L2 software prefetch operation.
  - Noncacheable hit error: This interrupt indicates that an access to a noncacheable area was found to be hit. This indicates user violation of a restriction, which obligates the user to flush the cache before changing descriptors' attributes.
  - Non-aligned non-allocate error: This interrupt indicates that a non-aligned access from the slave port isn't allocated in the cache and is forwarded to the external memory instead.
  - M2 non-mapped access error: This interrupt indicates that an access intended to access the L2 cache as M2, has exceeded the M2 boundaries indicating an issue with memory mapping to M2 configuration.
- Disabled on reset. Cannot be disabled after enabled.

11.5 M3 Memory

The 1056 KB M3 memory can be used for storing critical program and data shared between the cores and the device peripherals. The M3 memory has a 128-bit wide port and runs at 500 MHz. The M3 memory is ECC protected for soft errors.

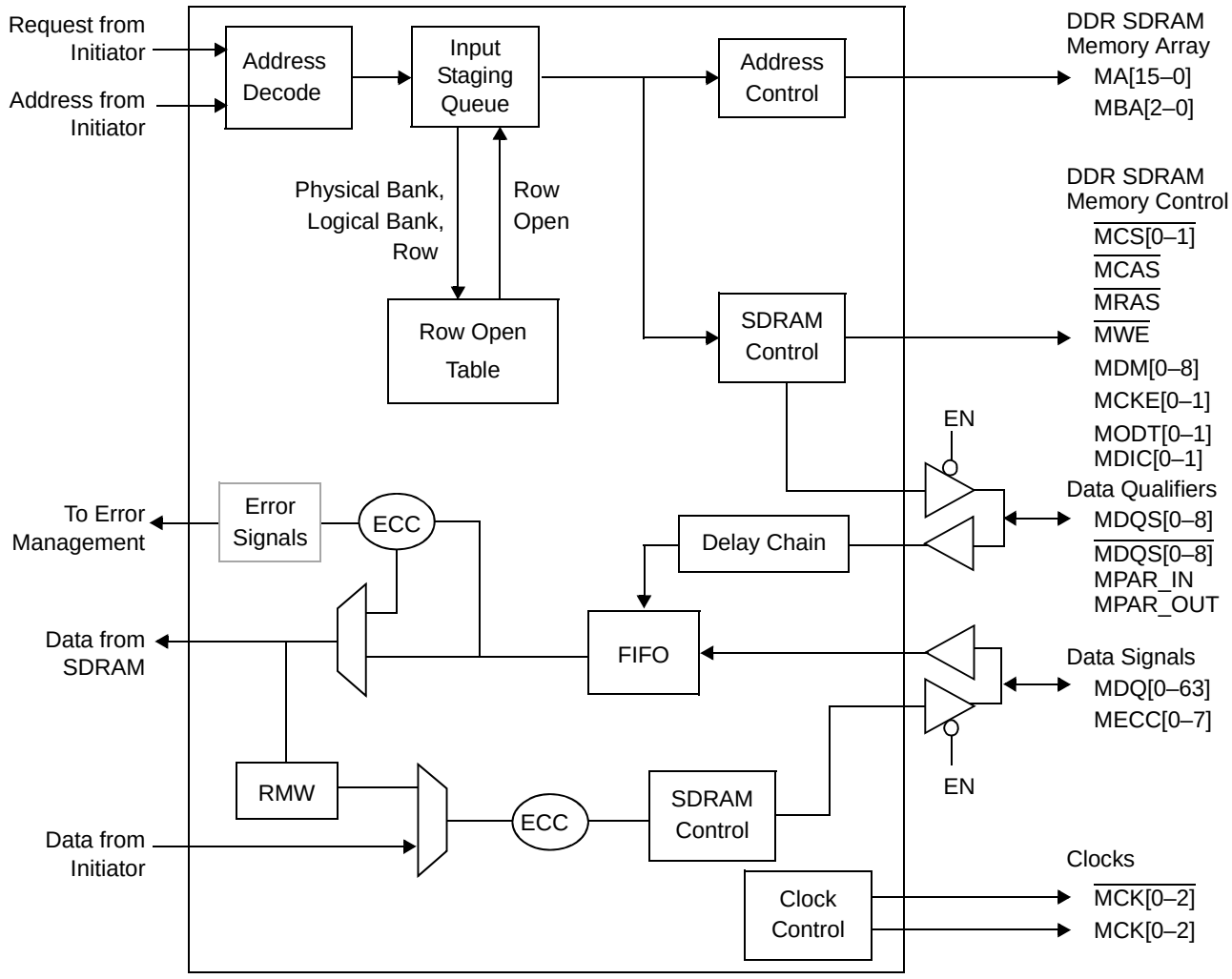
The M3 memory uses 64 KB compiled memory banks of SRAM. The memory is divided into three groups, two 512 KB groups and one 32 KB group, each with its own bus controller. The two 512 KB groups of memory are located between addresses 0xC0000000 and 0xC00FFFFF in the MSC8156E memory map. The 32 KB group is located between addresses 0xC0100000 and 0xC0107FFF. To support decreased power consumption, power to the two 512 KB memory groups can be disabled. All of the M3 memory supports semaphores and the RapidIO mail boxes. When the power is removed from the two 512 KB memory groups, the remaining 32 KB group still has power to provide minimal support for the hardware semaphores and the RapidIO mail boxes.

11.6 Internal Boot ROM

The MSC8156E device includes 96 KB of boot ROM accessible from all of the cores. This ROM provides the basic loading programming that allows the device to complete its initialization and load additional configuration and booting from external sources.

DDR SDRAM Memory Controller 12

The two fully programmable DDR SDRAM memory controllers support most available JEDEC-standard ×8 or ×16 DDR2 and DDR3 memories. 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. **Figure 12-1** shows a high-level view of the DDR memory controller with its associated interfaces.



Note: Register names in most of the chapter use the generic form shown in this figure. Actual names prepend M1 or M2 for the individual controllers.

Figure 12-1. DDR SDRAM Memory Controller Simplified Block Diagram

12.1 Features

Each one of the dual independent DDR memory controllers includes these distinctive features:

- Support for DDR2 and DDR3 SDRAM
- 64-/72-bit or 32-/40-bit SDRAM data bus for DDR2 and DDR3
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
  - Two physical banks (chip selects), each bank independently addressable
  - 256-Mbit to 2-Gbit devices depending on internal device configuration with ×8/×16 data ports (no direct ×4 support)
  - 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
- 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

The DDR memory controller supports the following modes:

- 32-byte cache line wrap.
- Dynamic power management mode. The DDR memory controller can reduce power consumption by deasserting 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].

12.2 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It supports JEDEC-compliant DDR2 and DDR3 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. In addition, DDR3 DIMM module specifications allow for vendors to use mirrored DIMMs, where some address and bank address lines are mirrored on the DIMM. The memory controller only supports these if the DDR_SDRAM_MD_CNTL register is used to initialize memory with DDR_SDRAM_CFG[BI] set. However, write leveling is not supported if these DIMMs are used.

Note: A bank is a physical bank specified by a chip select; a logical bank is one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the two or three bits on the memory bank address (MBA) pins during a memory access. Each memory interface supports two physical banks of 64-/72-bit or 32-/40-bit wide memory.

As shown in **Figure 12-1**, 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 two physical banks of 64-/72-bit wide or 32-/40bit wide memory. Total memory size can be up to 2 Gbyte while using one or two banks (chip selects).

Programmable parameters allow for a variety of memory organizations and timings. Using optional error checking and correcting (ECC) protection, the DDR memory controller detects and corrects all single-bit errors within the 64-bit or 32-bit data bus, detects all double-bit errors within the 64-bit or 32-bit data bus, and detects all errors within a nibble. The controller allows as many as 16 pages to be opened simultaneously. The amount of time (in clock cycles) the pages remain open is programmed via the DDR_SDRAM_INTERVAL[BSTOPRE] bit (see **Table 12-34** on page 12-74).

Read and write accesses to memory are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (four or eight) 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 with the ACTIVATE command specify 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, so the source of the data 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. These delays are implemented by the DDR SDRAM memory 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.

Figure 12-2 shows an example DDR SDRAM configuration with four logical banks.

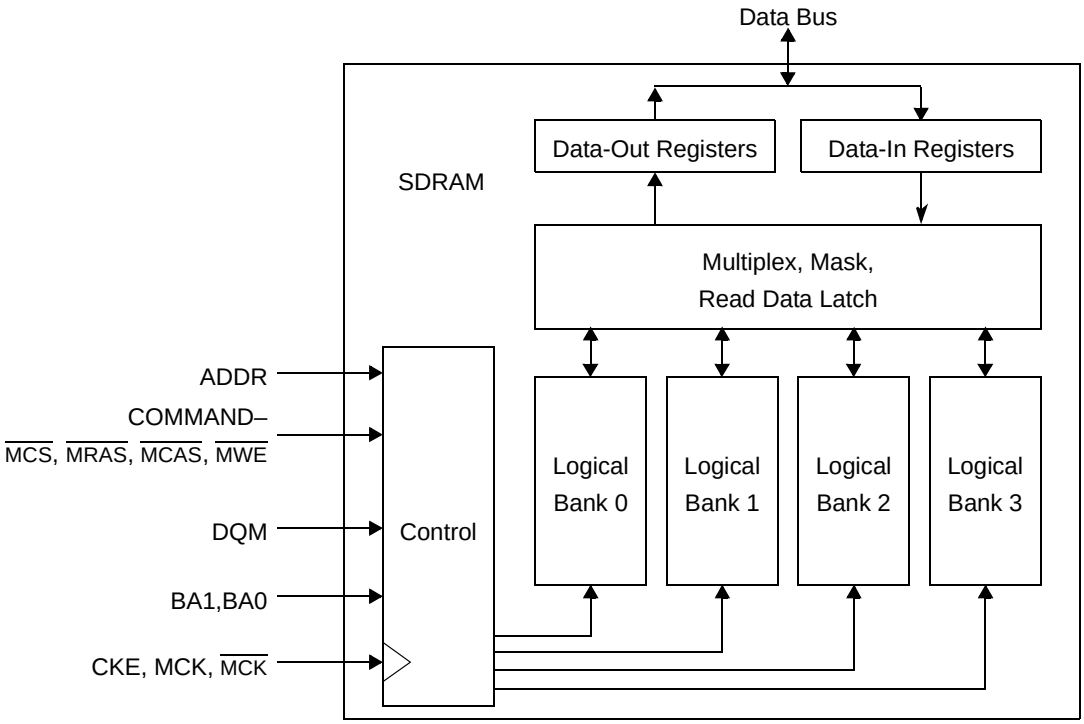


Figure 12-2. Typical Dual Data Rate SDRAM Internal Organization

Figure 12-3 shows some typical signal connections.

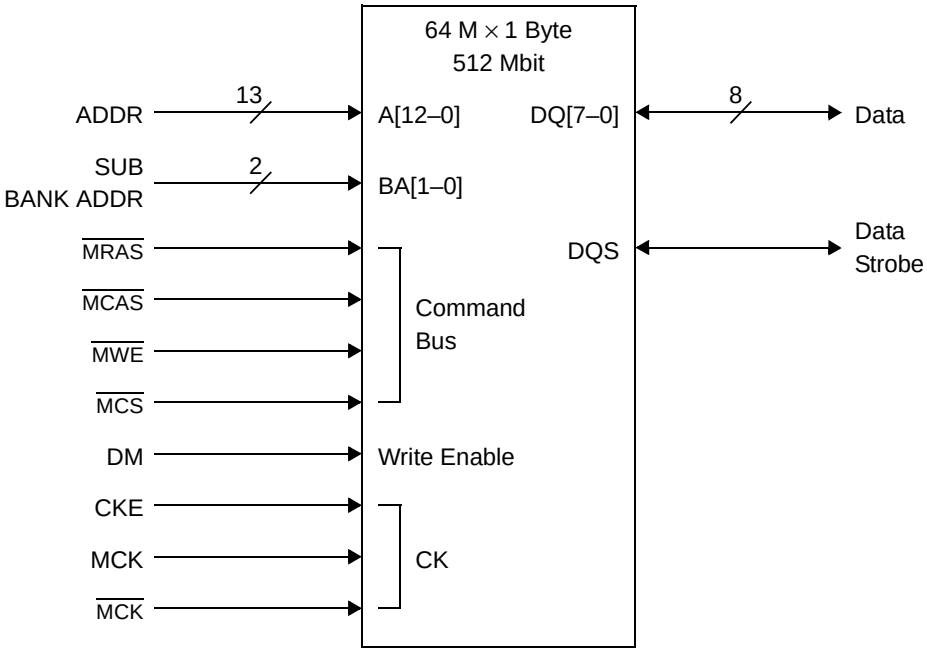
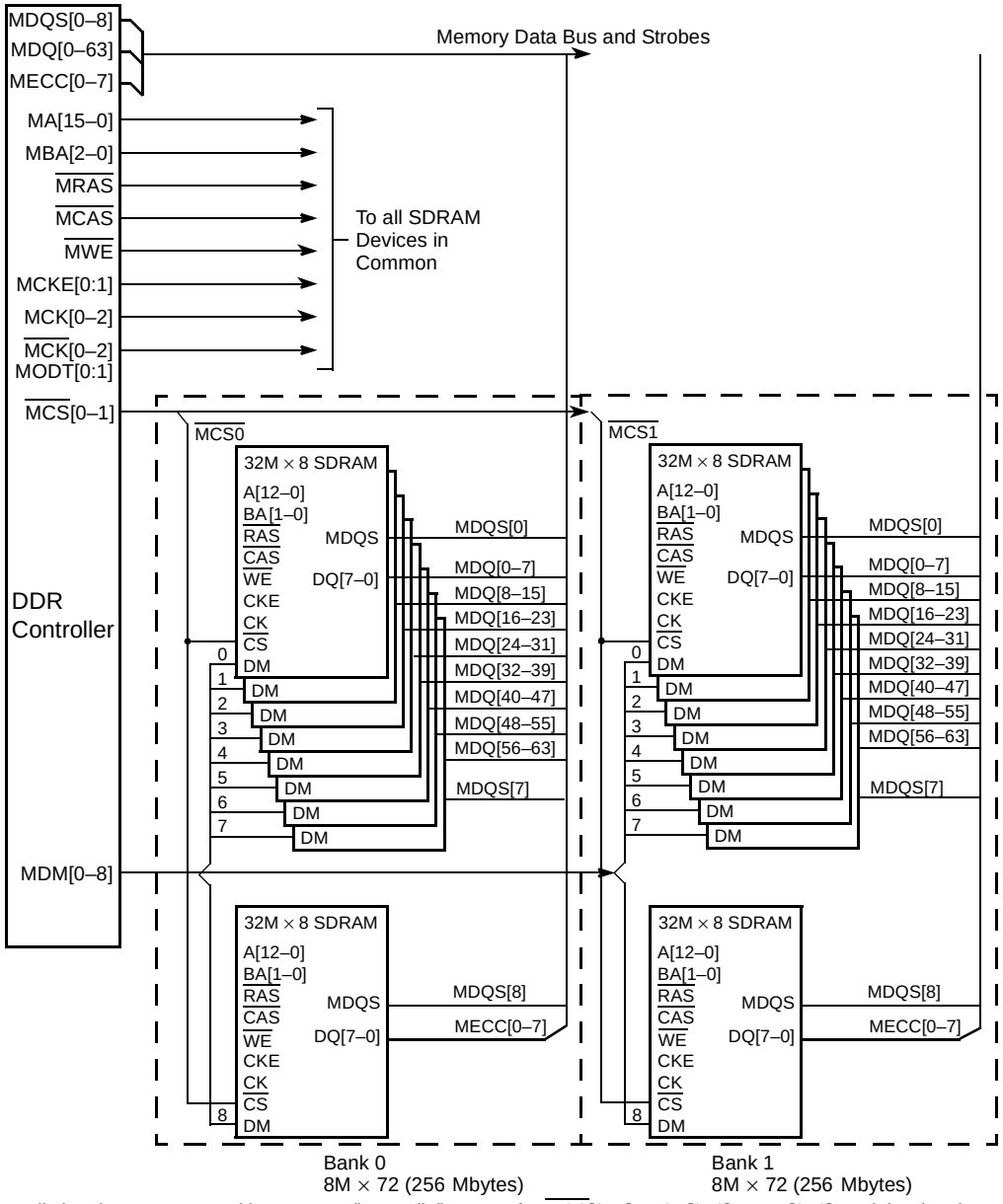


Figure 12-3. Typical DDR SDRAM Interface Signals

Figure 12-4 shows an example DDR SDRAM configuration with two physical banks, each containing nine 32M × 8 DDR modules for a total of 512 MB of system memory. One of the nine modules is used for the memory ECC checking function. Certain address and control lines software may require buffering. Analysis of the AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in defining signal buffering requirements. The DDR memory controller drives 16 address pins, but in this example the DDR SDRAM devices use only 13 bits.



1. All signals are connected in common (in parallel) except for MCS[0–1], MCK[0–2], MDM[0–8], and the data bus signals.
2. Each of the MCS[0–1] signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. MCK[0–2] can be apportioned among all memory devices. Complementary bus is not shown.

Figure 12-4. Example 512 MB DDR SDRAM Configuration With ECC

12.2.1 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. Sixteen multiplexed address signals MA[15–0] and three logical bank select signals MBA[2–0] support device densities from 256 Mb to 2 Gb. Two chip select signals $\overline{\text{MCS}}$[0–1] support one double rank DIMM or up to two single rank DIMM. 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 32 MB to 2 GB. The physical banks can be constructed using × 8 or ×16 memory devices. Supported memory technologies include 256 Mbits, 512 Mbits, 1 Gbit and 2 Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

Note: An 8-bit DDR SDRAM device has a DQM signal a pair of differential data strobe signals and eight data signals (DQ[0–7]). A 16-bit DDR SDRAM device has two DQM signals, two pairs of differential data strobe 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 64-bit 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. **Table 12-1** shows the relationship between data byte lane 0–7, MDM[0–7], MDQS[0–7] and MDQ[0–63] when DDR SDRAM memories are used with × 8 and × 16 devices.

Table 12-1. Byte Lane to Data Relationship

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 64-Bit Mode
0 (MSB)	MDM0	MDQS0	MDQ[0–7]
1	MDM1	MDQS1	MDQ[8–15]
2	MDM2	MDQS2	MDQ[16–23]
3	MDM3	MDQS3	MDQ[24–31]
4	MDM4	MDQS4	MDQ[32–39]
5	MDM5	MDQS5	MDQ[40–47]
6	MDM6	MDQS6	MDQ[48–55]
7 (LSB)	MDM7	MDQS7	MDQ[56–63]
8	MDM8	MDQS8	MECC[0–7]

Note: For a 32-bit interface, only data byte lanes 0, 1, 2, 3 and 8 (ECC) are used.

12.2.2 DDR SDRAM Organization

Although the DDR memory controller multiplexes row and column address bits onto 16 memory address signals MA[15:0] and 3 logical bank select signals MBA[2–0], individual physical banks can contain memory devices with fewer than 31 address bits. Each physical bank can be individually configured to provide from 12 to 16 row address bits plus 2 or 3 logical bank-select bits and from 8–11 column address bits. **Table 12-2** and **Table 12-3** list the DDR2 and DDR3 SDRAM device configurations supported by the DDR memory controller, respectively.

Note: DDR SDRAM is limited to 31 total address bits.

Table 12-2. Supported DDR2 Device Configurations

SDRAM Device	Device Configuration	Row × Column × Sub-bank Bits	64-Bit Bank Size	Two 64-bit Banks of Memory
256 Mb	32 M × 8	13 × 10 × 2	256 MB	512 MB
256 Mb	16 M × 16	13 × 9 × 2	128 MB	256 MB
512 Mb	64 M × 8	14 × 10 × 2	512 MB	1 GB
512 Mb	32 M × 16	13 × 10 × 2	256 MB	512 MB
1 Gb	128 M × 8	14 × 10 × 3	1 GB	2 GB
1 Gb	64 M × 16	13 × 10 × 3	512 MB	1 GB
2 Gb	256 M x 8	15 x 10 x 3	2 GB	NA
2 Gb	128 M × 16	14 × 10 × 3	1 GB	2 GB

Table 12-3. Supported DDR3 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	64-Bit Bank Size	Two 64-bit Banks of Memory
1 Gb	128 M × 8	14 × 10 × 3	1 GB	2 GB
1 Gb	64 M × 16	13 × 10 × 3	512 MB	1 GB
2 Gb	256 M x 8	15 x 10 x 3	2 GB	NA
2 Gb	128 M x 16	14 x 10 x 3	1 GB	2 GB

Note: DDR SDRAM is limited to 31 total address bits.

Table 12-4. Supported DDR2 Device Configurations

SDRAM Device	Device Configuration	Row × Column × Sub-bank Bits	32-Bit Bank Size	Two 32-bit Banks of Memory
256 Mb	32 M × 8	13 × 10 × 2	128 MB	256 MB
256 Mb	16 M × 16	13 × 9 × 2	64 MB	128 MB
512 Mb	64 M × 8	14 × 10 × 2	256 MB	512 MB
512 Mb	32 M × 16	13 × 10 × 2	128 MB	256 MB
1 Gb	128 M × 8	14 × 10 × 3	512 MB	1 GB

Table 12-4. Supported DDR2 Device Configurations

SDRAM Device	Device Configuration	Row × Column × Sub-bank Bits	32-Bit Bank Size	Two 32-bit Banks of Memory
1 Gb	64 M × 16	13 × 10 × 3	256 MB	512 MB
2 Gb	256 M × 8	15 × 10 × 3	1 GB	2 GB
2 Gb	128 M × 16	14 × 10 × 3	512 MB	1 GB

Table 12-5. Supported DDR3 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	32-Bit Bank Size	Two 32-bit Banks of Memory
1 Gb	128 M × 8	14 × 10 × 3	512 MB	1 GB
1 Gb	64 M × 16	13 × 10 × 3	256 MB	512 MB
2 Gb	256 M × 8	15 × 10 × 3	1 GB	2 GB
2 Gb	128 M × 16	14 × 10 × 3	512 MB	1 GB

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 (see **Section 12.6**, *Error Management*, on page 12-35).

If the starting and ending address of a disabled bank overlaps with the address space of an enabled bank, system memory in the overlapping address range can be corrupted. The starting and ending addresses of unused memory banks should be mapped to unused memory space.

Using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware configures the memory-boundary registers to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate $\overline{\text{MCSx}}$ signal for memory accesses according to the provided bank starting and ending addresses. The memory banks do not have to be mapped to a contiguous address space.

12.2.3 DDR SDRAM Address Multiplexing

Table 12-6 and **Table 12-7** show the address bit encodings for each DDR SDRAM configuration. The address at the memory controller signals MA[15–0] use MA15 as the MSB and MA0 as the LSB. MA10 is the auto-precharge bit in DDR2/DDR3 modes for reads and writes, so the column address can never use MA10.

Table 12-6. DDR2 Address Multiplexing for 64-bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		MSB		Address from Core Initiator																										LSB
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2–0
15 x 10 x 3	$\overline{MRAS}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{MRAS}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{MRAS}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{MRAS}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{MRAS}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{MRAS}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	$\overline{MBA}$																			1	0										
	$\overline{MCAS}$																					8	7	6	5	4	3	2	1	0	

Table 12-7. DDR2 Address Multiplexing for 32-bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		MSB		Address from Core Initiator																									LSB
		30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
15 x 10 x 3	$\overline{MRAS}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{MRAS}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{MRAS}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{MRAS}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	$\overline{MBA}$																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{MRAS}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	$\overline{MBA}$																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{MRAS}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	$\overline{MBA}$																			1	0										
	$\overline{MCAS}$																					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 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.

Table 12-8 and **Table 12-9** illustrate examples of address decode when interleaving between two chip selects,

Table 12-8. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Disabled

Row x Col		MSB		Address from Core Initiator																									LSB
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2–0
15 x 10 x 3	$\overline{MRAS}$	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																														
	$\overline{MCAS}$																														
14 x 10 x 3	$\overline{MRAS}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{MRAS}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL													
	MBA																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{MRAS}$			12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																	2	1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{MRAS}$				12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL													
	MBA																		1	0											
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0	

Table 12-9. Example of Address Multiplexing for 32-Bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Disabled

Row x Col		MSB			Address from Core Initiator																										LSB
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
15 x 10 x 3	$\overline{MRAS}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																															
	$\overline{MCAS}$																															
14 x 10 x 3	$\overline{MRAS}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																	2	1	0												
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0		
14 x 10 x 2	$\overline{MRAS}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL													
	MBA																		1	0												
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0		
13 x 10 x 3	$\overline{MRAS}$				12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																	2	1	0												
	$\overline{MCAS}$																				9	8	7	6	5	4	3	2	1	0		
13 x 10 x 2	$\overline{MRAS}$					12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL													
	MBA																			1	0											
	$\overline{MCAS}$																				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 are swapped, and the sub-bank is decoded as the most significant portion of the DRAM address, as shown in **Table 12-10**.

If chip select interleaving and PASR are enabled simultaneously for a chip select, then the interleaved chip select bit is placed immediately at the top the left of column decode as shown in **Table 12-11**.

Table 12-10. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled for 32-Bit Data Bus Without Interleaving

Row x Col		msb	Address from Core Initiator																												lsb
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1–0
15 x 10 x 3	MRAS						14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA			2	1	0																										
	MCAS																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	MRAS							13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA				2	1	0																									
	MCAS																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	MRAS							13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA					1	0																									
	MCAS																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	MRAS								12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA					2	1	0																								
	MCAS																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	MRAS								12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA						1	0																								
	MCAS																					9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	MRAS									12	11	10	9	8	7	6	5	4	3	2	1	0										
	MBA							1	0																							
	MCAS																						8	7	6	5	4	3	2	1	0	

Table 12-11. 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 Initiator																											lsb
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2–0
15 x 10 x3	MRAS				14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA	2	1	0																CS SEL												
	MCAS																															
14 x 10 x3	MRAS					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA		2	1	0															CS SEL												
	MCAS																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	MRAS					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA			1	0															CS SEL												
	MCAS																					9	8	7	6	5	4	3	2	1	0	

Table 12-11. Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled (Continued)

Row x Col		msb		Address from Core Initiator																								lsb
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2–0
13 x 10 x 3	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL											
	MBA			2	1	0																									
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL											
	MBA				1	0																									
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

12.3 JEDEC Standard DDR SDRAM Interface Commands

This section describes the commands and timings for DDR2 or DDR3 modes. The DDR memory controller performs all read or write accesses to DDR SDRAM 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 on both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

Following are the DDR SDRAM interface commands (summarized in **Table 12-12** and **Table 12-14**) 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 $\overline{\text{MCAS}}$ before $\overline{\text{MRAS}}$). 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 it is read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before a refresh. The memory controller also supports posted refreshes in which several refreshes execute at once, and the refresh interval can be extended.

- *Mode register set (for configuration)*. Allows DDR SDRAM options to be set. These options are: $\overline{\text{MCAS}}$ latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length. $\overline{\text{MCAS}}$ latency can be chosen as provided by the preferred SDRAM (some SDRAMs provide $\overline{\text{MCAS}}$ latency {1,2,3}, some provide $\overline{\text{MCAS}}$ latency {1,2,3,4,5}, and so on). Burst type is always sequential. Although some SDRAMs provide burst lengths of 4 and 8, this memory controller supports only a burst length of 4. A burst length of 8 is supported only for DDR3 memory devices. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller. The DDR memory controller executes the mode register set command during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ 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[MEMEN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, software can configure the mode registers via the DDR_SDRAM_MD_CNTL register.
- S*elf refresh*. For use 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 places all logical bank in a precharged state.

Table 12-12. DDR2 Command Truth Table

Function	CKE		$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA0 BA1 BA2	A15–A11	A10	A9– A0	Notes
	Previous Cycle	Current Cycle									
(Extended) Mode Register Set (MRS)	H	H	L	L	L	L	BA	OP Code			1, 2
Refresh (REF)	H	H	L	L	L	H	X	X	X	X	1
Self Refresh Entry (SRE)	H	L	L	L	L	H	X	X	X	X	1, 8
Self Refresh Exit (SRX)	L	H	H	X	X	X	X	X	X	X	1, 7, 8
			L	H	H	H					
Single Bank Precharge (PRE)	H	H	L	L	H	L	BA	X	L	X	1, 2
Precharge All Banks (PREA)	H	H	L	L	H	L	X	X	H	X	1
Bank Activate (ACT)	H	H	L	L	H	H	BA	Row Address			1, 2
Write (WR)	H	H	L	H	L	L	BA	Column	L	Column	1, 2, 3
Write with auto-precharge (WRA)	H	H	L	H	L	L	BA	Column	H	Column	1, 2, 3
Read (RD)	H	H	L	H	L	H	BA	Column	L	Column	1, 2, 3
Read with auto-precharge (RDA)	H	H	L	H	L	H	BA	Column	H	Column	1, 2, 3
No operation (NOP)	H	X	L	H	H	H	X	X	X	X	1
Device Deselect (DES)	H	X	H	X	X	X	X	X	X	X	1
Power Down Entry (PDE)	H	L	H	X	X	X	X	X	X	X	1, 4
			L	H	H	H					
Power Down Exit (PDX)	L	H	H	X	X	X	X	X	X	X	1, 4
			L	H	H	H					

Notes:
1. All DDR2 SDRAM commands are defined by states of $\overline{CS}$, $\overline{RAS}$, $\overline{WE}$, and CKE at the rising edge of the clock.
2. Bank addresses BA0, BA1, BA2 (BA) determine which bank is operated on. For (E)MRS BA selects an (Extended) Mode Register.
3. Burst reads or writes at BL = 4 cannot be terminated or interrupted. See “Reads interrupted by a Read” and “Writes interrupted by a Write” in **section 2.6** of the *JEDEC DDR2 SDRAM Specification* (JESD79-2C).
4. The Power Down Mode does not perform any refresh operations. The during of Power Down is therefore limited by the refresh requirements listed in **section 2.9** of the *JEDEC DDR2 SDRAM Specification* (JESD79-2C)..
5. The state of ODT does not affect the state described in this table. The ODT function is not available during Self-Refresh. See. **section 2.4.4** of the *JEDEC DDR2 SDRAM Specification* (JESD79-2C).
6. X can be H or L, but must be a defined logic level.
7. Self refresh exit is asynchronous.
8. VREF must be maintained during Self Refresh operation.

Table 12-13. DDR3 Command Truth Table

Function	CKE		$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA0 BA1 BA2	A15–A13	A12 $\overline{BC}$	A10	A9–A0 A11	Notes
	Previous Cycle	Current Cycle										
Mode Register Set (MRS)	H	H	L	L	L	L	BA	OP Code				
Refresh (REF)	H	H	L	L	L	H	V	V	V	V	V	
Self Refresh Entry (SRE)	H	L	L	L	L	H	V	V	V	V	V	7, 9, 12
Self Refresh Exit (SRC)	L	H	H	V	V	V	V	V	V	V	V	7, 8, 9, 12
			L	H	H	H						
Single Bank Precharge (PRE)	H	H	L	L	H	L	BA	V	V	L	V	
Precharge All Banks (PREA)	H	H	L	L	H	L	V	V	V	H	V	
Bank Activate (ACT)	H	H	L	L	H	H	BA	Row Address				1, 2
Write (Fixed BL8 or BC4) (WR)	H	H	L	H	L	L	BA	RFU	V	L	CA	
Write (BC4, on the Fly) (WRS4)	H	H	L	H	L	L	BA	RFU	L	L	CA	
Write (BC8, on the Fly) (WRS8)	H	H	L	H	L	L	BA	RFU	H	L	CA	
Write with auto-precharge (Fixed BL8 or BC4) (WRA)	H	H	L	H	L	L	BA	RFU	V	H	CA	
Write with auto-precharge (BC4, on the Fly) (WRAS4)	H	H	L	H	L	L	BA	RFU	L	H	CA	
Write with auto-precharge (BL8) (WRAS8)	H	H	L	H	L	L	BA	RFU	H	H	CA	
Read (Fixed BL8 or BC4) (RD)	H	H	L	H	L	H	BA	RFU	V	H	CA	
Read (BC4, on the Fly) (RDS4)	H	H	L	H	L	H	BA	RFU	L	H	CA	
Read (BL8, on the Fly) (RDS8)	H	H	L	H	L	H	BA	RFU	H	H	CA	
Read with auto-precharge (Fixed BL8 or BC4) (RDA)	H	H	L	H	L	H	BA	RFU	V	H	CA	
Read with auto-precharge (BC4, on the Fly) (RDAS4)	H	H	L	H	L	H	BA	RFU	L	H	CA	

Table 12-13. DDR3 Command Truth Table (Continued)

Function	CKE		$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA0 BA1 BA2	A15–A13	A12 $\overline{BC}$	A10	A9–A0 A11	Notes
	Previous Cycle	Current Cycle										
Read with auto-precharge (BL8, on the Fly) (RDAS8)	H	H	L	H	L	H	BA	RFU	H	H	CA	
No operation (NOP)	H	H	L	H	H	H	V	V	V	V	V	10
Device Deselected (DES)	H	H	H	X	X	X	X	X	X	X	X	11
Power Down Entry (PDE)	H	L	L	H	H	H	V	V	V	V	V	6. 12
			H	V	V	V						
Power Down Exit (PDX)	L	H	L	H	H	H	V	V	V	V	V	1, 4
			H	V	V	V						
ZQ Calibration Long (ZQCL)	H	H	L	H	H	L	X	X	X	H	X	

Notes:
1. All DDR3 SDRAM commands are defined by states of $\overline{CS}$, $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, and CKE at the rising edge of the clock. The MSB of BA, RA, and CA are device density and configuration dependent.
2. $\overline{RESET}$ is Low enable command which is only used for asynchronous reset and must be maintained HIGH during any function.
3. Bank addresses BA0, BA1, BA2 (BA) determine which bank is operated on. For (E)MRS BA selects an (Extended) Mode Register.
4. V means H or L, but must be a defined level. X means either defined or undefined (such as floating) logic level.
5. Burst reads or writes cannot be terminated or interrupted. Fixed/on-the-Fly BL is defined by MRS..
6. The Power Down Mode does not perform any refresh operations.
7. The state of ODT does not affect the states described in this table. The ODT function is not available during Self-Refresh.
8. Self refresh exit is asynchronous.
9. VREF (both VrefDQ and VrefCA) must be maintained during Self-Refresh operation.
10. The No Operation command should be used in cases when th DDR3 SDRAM is in an idle or wait state. The purpose of the NOP command is to prevent the DDR3 SDRAM from registering any unwanted commands between operations. A NOP does not terminate a previous operation that is still executing, such as a burst read or write cycle.
11. The Deselect command perfoms the same function as a NOP.
12. Refer to the CKE Truth Table in the *JEDEC DDR3 SDRAM Specification* (JESD79-3B)for details about CKE transition.

12.4 DDR SDRAM Clocking and Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four-beat burst read but ignores the last three beats. Single-beat writes are performed by masking the last three beats of the four-beat burst using the data mask MDM[0–8]. If ECC is disabled, writes smaller than the data bus width 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 you to set the timing intervals listed in **Table 12-14** with a granularity of 1, 1/2, 1/4, or 1/8 SDRAM clock cycle. The value of these parameters (in whole clock cycles) must be set by application software at system initialization before the DDR controller is enabled and must be kept in the DDR memory controller configuration registers. Any update of the timing parameters should be done while the controller is disabled.

Table 12-14. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition	Register/Page
ACTTOACT	**Activate-to-Activate Interval** The number of clock cycles from a bank-activate command to another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-55**
ACTTOPRE	**Activate-to-Precharge Interval** 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	**Activate-to-Read/Write Interval for SDRAM** 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	**Open Page Interval** 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.	DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL) **page 12-74**
CASLAT	**MCAS Latency from READ Command** 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 $n + m$.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-55**
PRETOACT	**Precharge-to-Activate Interval** 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}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-55**

Table 12-14. DDR SDRAM Interface Timing Intervals (Continued)

Timing Intervals	Definition	Register/Page
REFINT	**Refresh Interval** Represents the number of memory bus clock cycles between refresh cycles. One row is refreshed in each SDRAM bank during each refresh cycle. 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. This interval is listed in the AC specifications of the SDRAM as t_{REFI}.	DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL) **page 12-74**
REFREC	**Refresh Recovery Time** The number of clock cycles from the refresh command until an activate command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RFC}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-55** and DDR SDRAM Timing Configuration Register 3 (TIMING_CFG_3) **page 12-50**
WR_DATA_DELAY	**Write 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,	DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) **page 12-60**
WRREC	**Write Recovery** 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}.	DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1) **page 12-55**
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. This interval is listed in the AC specifications of the SDRAM as t_{WTR}.	
CLK_ADJUST	**Clock Adjust** Adjusts the MCK timing relative to address/command. The delay is set from the address/command start timing to the MCK rising edge. The resolution is 1/8 SDRAM clock cycle.	DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL) **page 12-75**
ADD_LAT	**Additive Latency** Specifies the number of clock cycles for a Posted CAS operation until the registered read/write command is issued inside the SDRAM as defined in the JEDEC DDR2 and DDR3 standards.	DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) **page 12-60**
CPO (if automatic calibration is not selected)	**CAS-to-Preamble Override** Specifies when the DDR controller starts waiting for the first DQS rising edge from DDR memory during the DQS preamble for read access. The DQS rising edge should occur within 1 clock cycle from the timing defined by this parameter.	DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) **page 12-60**

Table 12-14. DDR SDRAM Interface Timing Intervals (Continued)

Timing Intervals	Definition	Register/Page
WR_LAT	**Write Latency** Specifies the number of clock cycles between the WRITE command and the first data write when AL = 0. When additive latency is applied, WR_LAT specifies the number of clock cycles between issuing the REGISTERED WRITE command inside the SDRAM and the first data write.	DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) **page 12-60**
RD_TO_PRE	Read to Precharge (tRTP) Specifies the number of clock cycles between the READ command and the PRECHARGE command when AL = 0. When additive latency is applied, RD_TO_PRE specifies the minimum tRTP timing from the REGISTERED READ command inside the SDRAM and the PRECHARGE command.	DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2) **page 12-60**

Software should initialize the parameters in the DDR controller registers with the appropriate values before the controller is enabled. Altering the register values while the controller is enabled can produce unpredictable controller behavior, can cause data loss, and can lock the controller or device. System software is responsible for optimally configuring of 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 attempting any accesses to SDRAM.

Figure 12-5 through **Figure 12-7** show DDR SDRAM timing for various types of accesses, including a single-beat read, a single-beat write, and a burst-write. Note that all control signals transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads also note that data and MDQS signal transitions occur on every edge of the memory bus clock. **Figure 12-5** through **Figure 12-7** assume that DDR_SDRAM_CLK_CNTL[CLK_ADJUST] = 0100 (set to 1/2 SDRAM cycle), the additive latency is 0 SDRAM cycles, and the write latency is 1 SDRAM cycle.

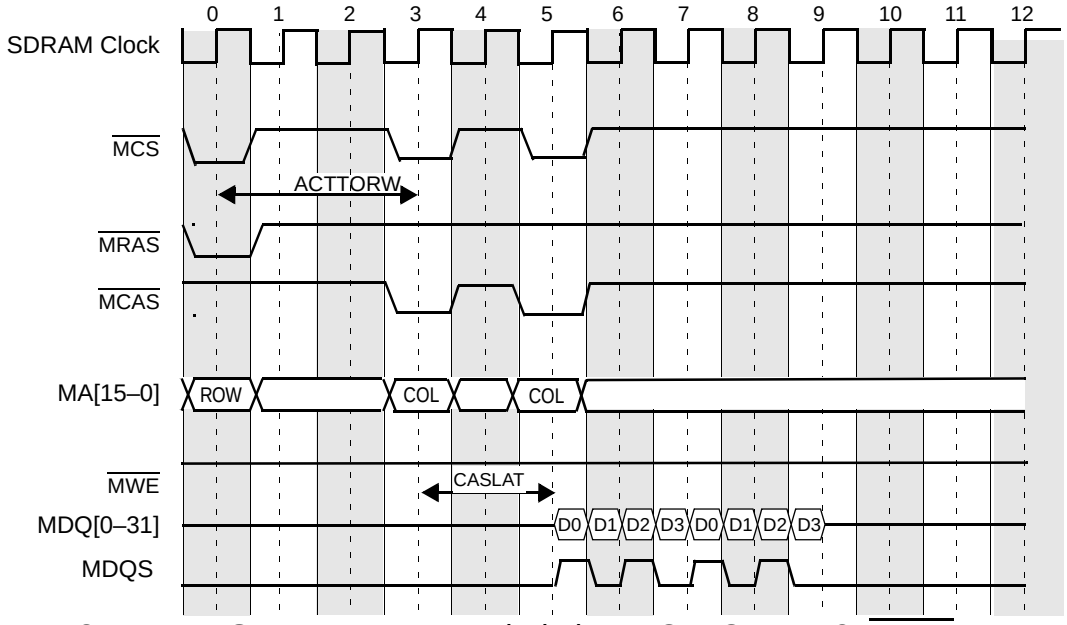


Figure 12-5. DDR SDRAM Burst Read Timing: ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

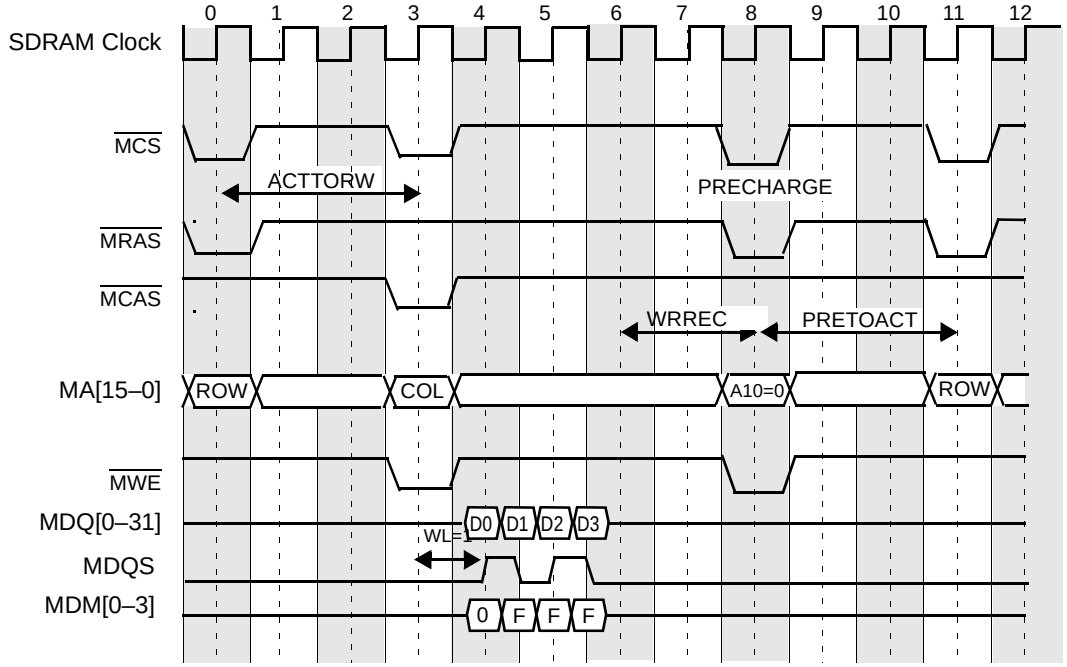


Figure 12-6. DDR SDRAM Single-Beat (32-Bit) Write Timing: ACTTORW = 3

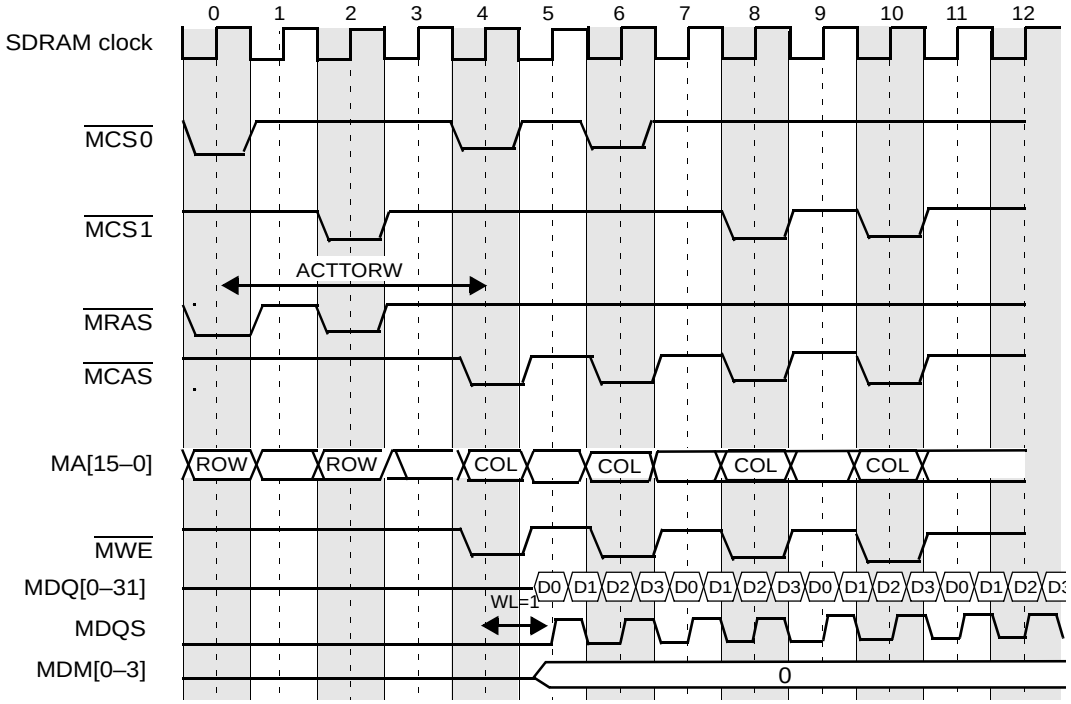


Figure 12-7. DDR SDRAM 4-Beat Burst Write Timing: ACTTORW = 4

12.4.1 Clock Distribution

- If a system is composed of many devices, consider using zero-delay PLL clock buffers that are compliant with the JEDEC-JESD82 standard. These buffers are designed for DDR applications.
- A 72-bit × 64 Mbyte DDR bank has 9-byte-wide DDR memory chips resulting in 18 DDR chips in a two-bank system. For this case, each MCK/$\overline{\text{MCK}}$ signal pair should drive exactly six devices, as shown in **Figure 12-8**.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide details on PCB layout and termination issues.

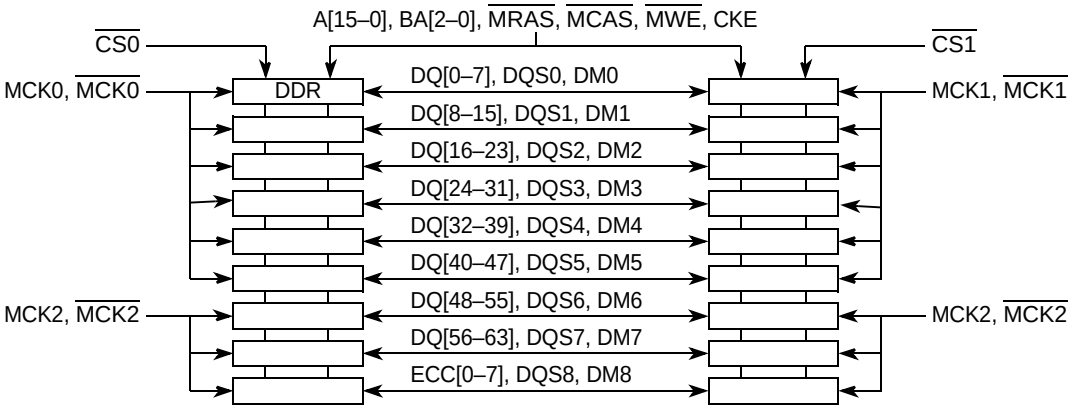


Figure 12-8. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

12.4.2 DDR SDRAM Mode-Set Command Timing

The DDR 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. **Figure 12-9** shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE of DDR_SDRAM_MODE in the case of automatic hardware initialization. The Mode Register Set cycle time is set to 2 DRAM cycles in **Figure 12-9**.

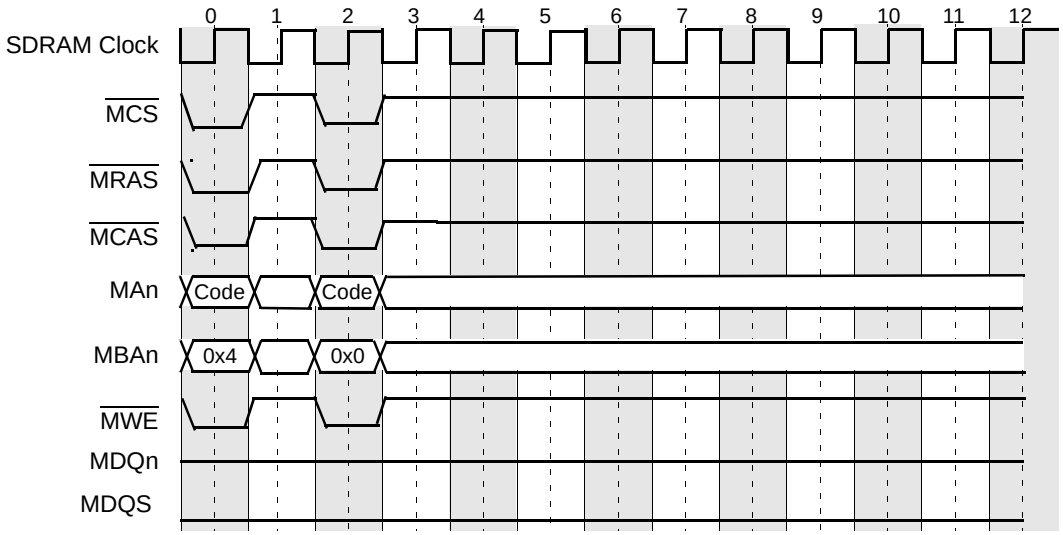


Figure 12-9. DDR SDRAM Mode-Set Command Timing

12.4.3 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 DIMM 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.

Figure 12-10 shows the registered DDR SDRAM DIMM single-beat write timing.

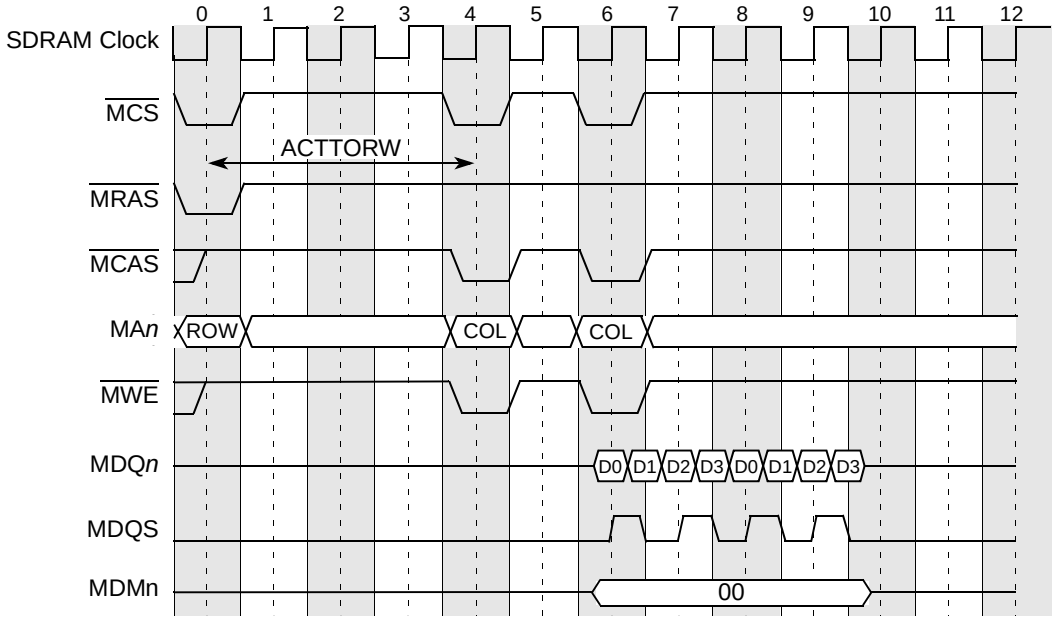


Figure 12-10. Registered DDR SDRAM DIMM Burst Write Timing

12.4.4 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, DATA DELAY, configured in 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 TIMING_CFG_2[WR_DATA_DELAY] parameter can be used to meet this timing requirement for a variety of system configurations, ranging from a system with one bank of SDRAM devices to a fully populated system.
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.

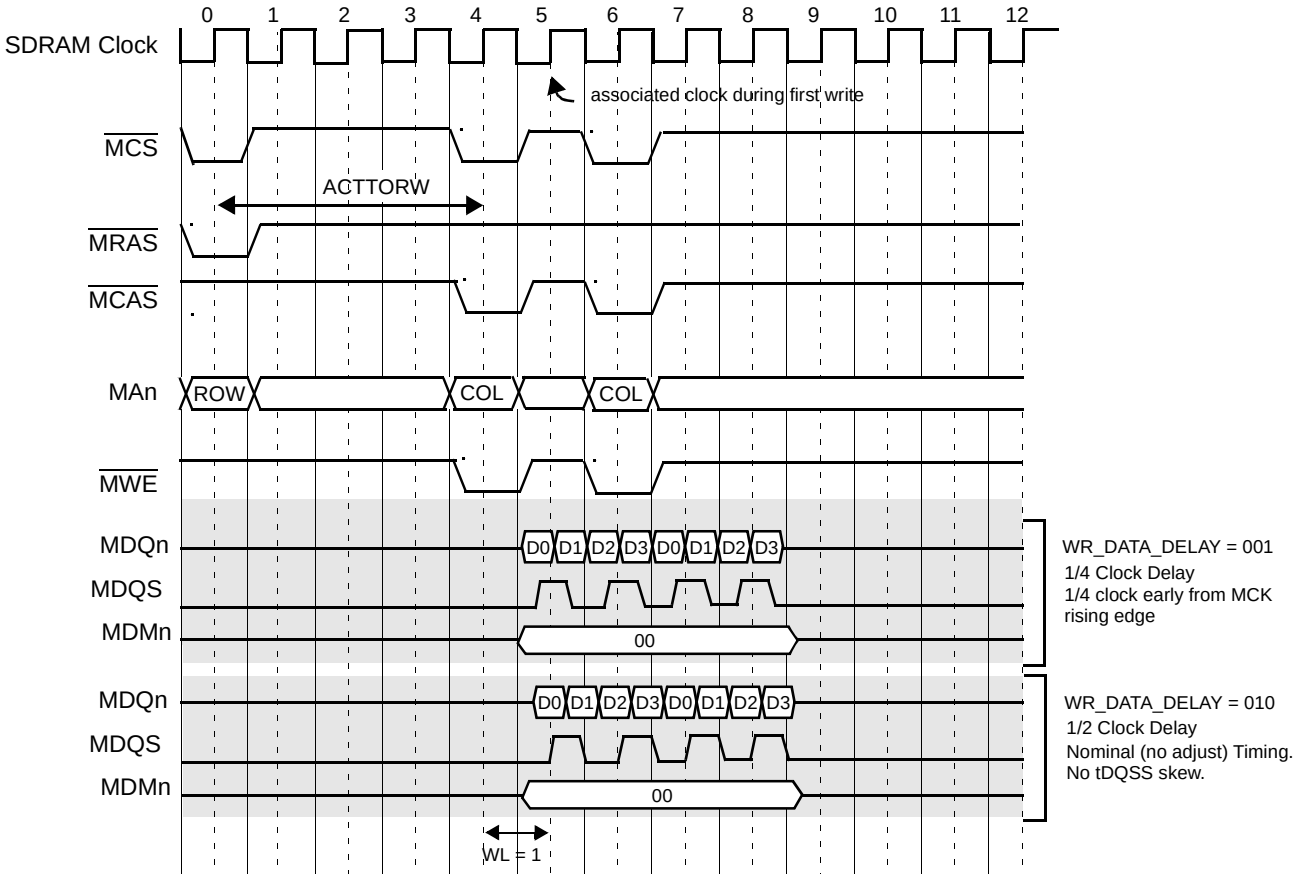


Figure 12-11. Write Timing Adjustments Example for Write Latency = 1 for the Case Where DDR_SDRAM_CLK_CNTL[CLK_ADJUST] = 0100

12.4.5 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 any 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 possible banks to reduce instantaneous power requirements. Three sets of auto refresh commands are issued on consecutive cycles. The initial PRECHARGE ALL commands are also staggered in three groups. When the system enters self refresh mode, only one refresh command is issued simultaneously to all physical banks. For the entire refresh sequence, no cycle optimization occurs for the usual case where fewer 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 in DDR_SDRAM_CFG_2[NUM_PR] allow the refresh interval to be set to a larger value.

Note: The MSC8156E initiates three cycles of PRECHARGE ALL commands and three cycles of REFRESH commands although there are only two banks (chip selects) available,

12.4.5.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1[REFREC] and TIMING_CFG_3[EXT_REFREC], which specify 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 12-12** (TIMING_CFG_1[REFREC] = 10 in this example). System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and

TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be complete before DDR SDRAM accesses are attempted.

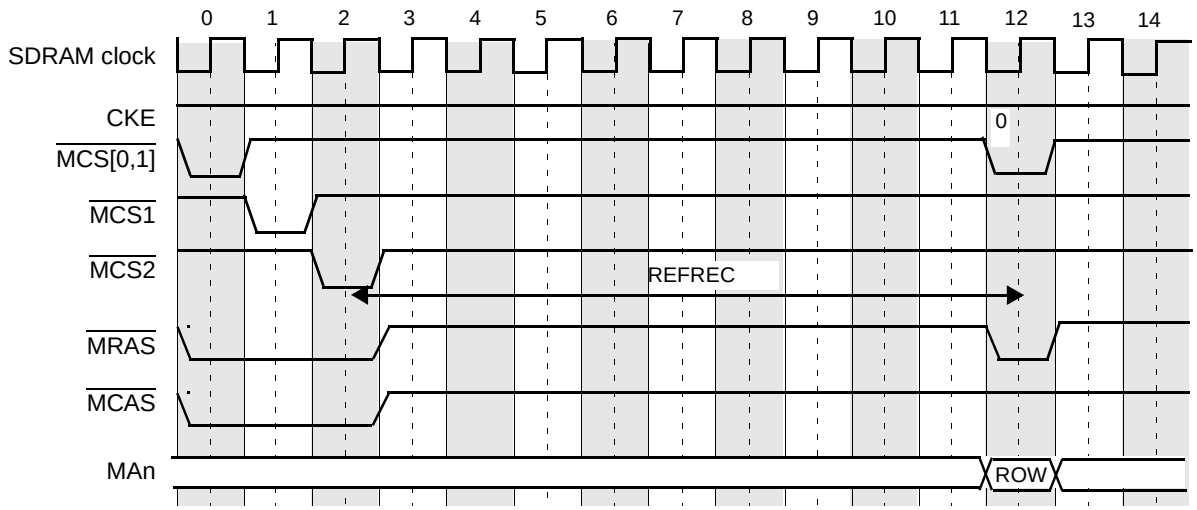


Figure 12-12. DDR SDRAM Bank Staggered Auto Refresh Timing

12.4.5.2 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 by the DDR_SDRAM_CFG[SREN] bit.

Note: In absence of self-refresh support, system software must preserve the DDR's DRAM data (for example, by copying the memory content to a non-volatile memory, such as a disk, Flash memory, and so on) before entering power saving mode.

The dynamic power-saving mode uses the CKE pin to power down the memory device dynamically when there is no system memory activity. The CKE pin is deasserted when both conditions are met

- No memory refreshes are scheduled.
- No memory accesses are scheduled.

The CKE pin 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].

Dynamic power management mode offers tight control of the memory system 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 upon whether active or precharge power-down is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of one cycle is shown in **Figure 12-13**.

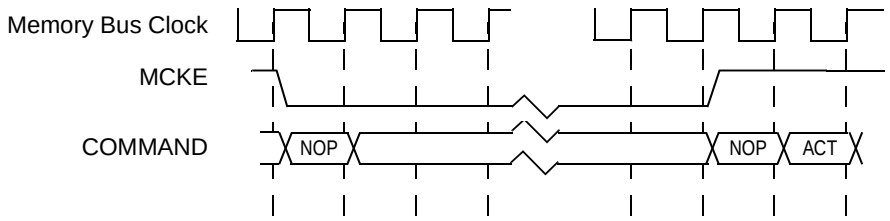


Figure 12-13. DDR SDRAM Power-Down Mode

The entry and exit timing for self-refreshing SDRAMs in Sleep mode is shown in **Figure 12-14** and **Figure 12-15**.

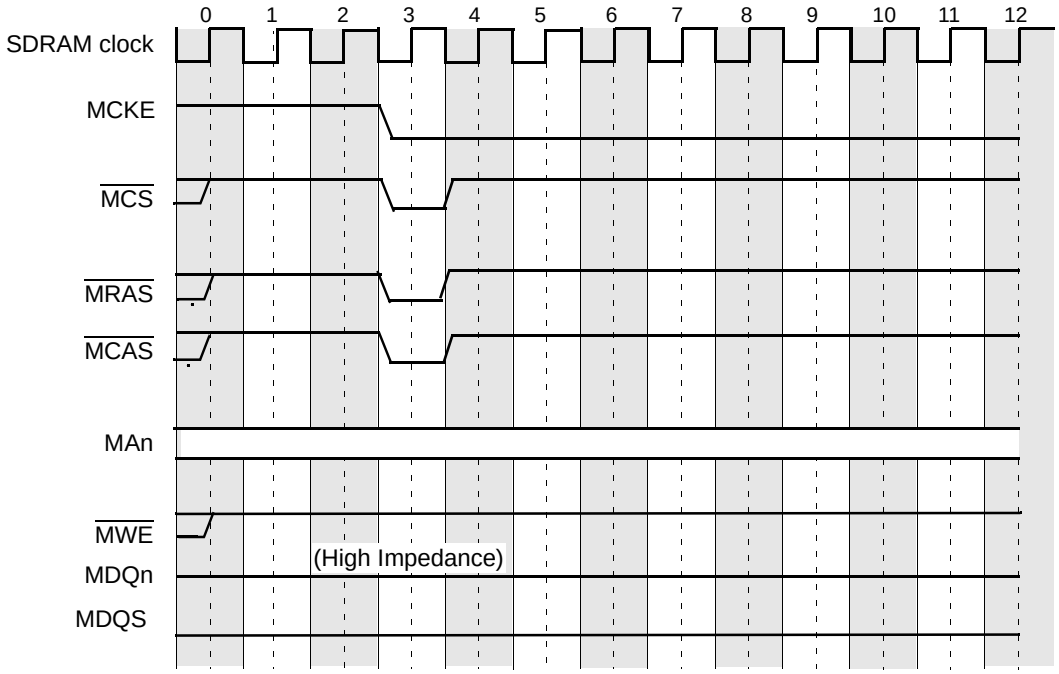


Figure 12-14. DDR SDRAM Self-Refresh Entry Timing

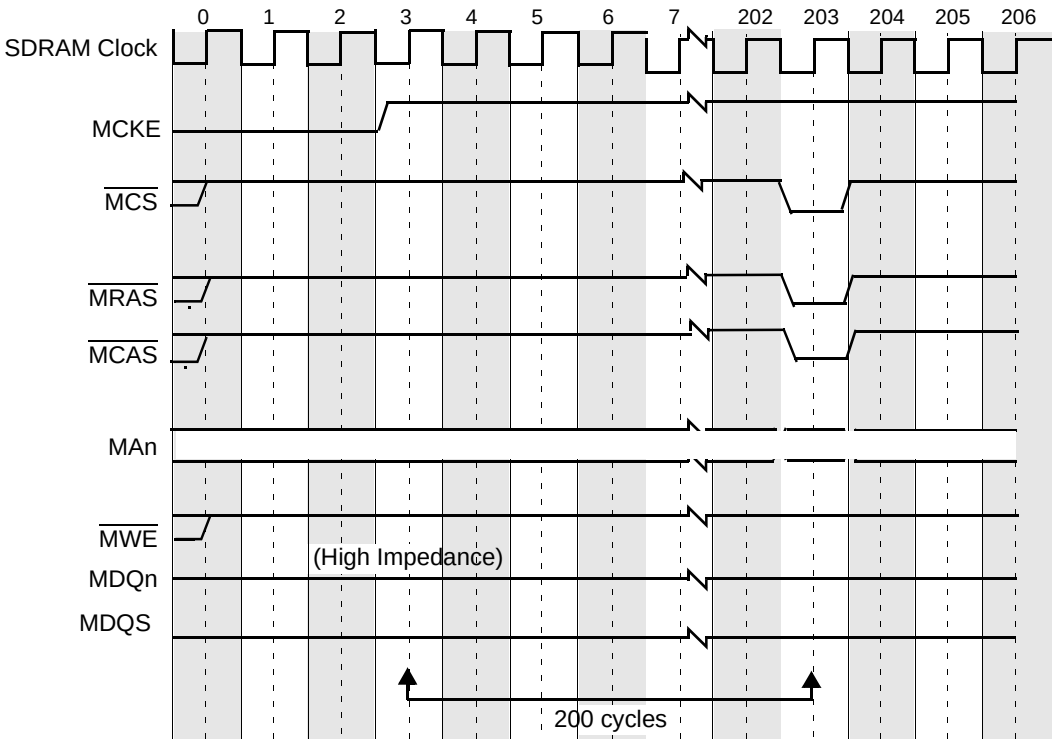


Figure 12-15. DDR SDRAM Self-Refresh Exit Timing

12.4.6 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes for a 64-bit bus). For transfer sizes other than four or eight beats, the data transfers are still in four- or eight-beat bursts. If ECC is enabled and either the access is not 64-bit-aligned or the size is not a multiple of 64 bits, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or the access is 64-bit-aligned with a size that is a multiple of 64 bits, the data masks MDM[0–8] (MDM[0–3,8] for a 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 64-bit words from writing to SDRAM. For example, if a write transaction with a size of 8 bytes is desired, then the second, third, and fourth beats of data are not written to SDRAM.

Table 12-15 lists the data beat sequencing to and from the DDR SDRAM and the data queues for each of the possible transfer sizes with each of the possible starting 64-bit offsets. All underlined 64-bit offsets are valid for the transaction.

Table 12-15. Memory Controller–Data Beat Ordering for the 64-Bit Interface

Transfer Size	Starting Double-Word Offset	Double-Word Sequence to/from DRAM and Queues
8 bytes	0	**0** - 1 - 2 - 3
	1	**1** - 2 - 3 - 0
	2	**2** - 3 - 0 - 1
	3	**3** - 0 - 1 - 2
16 bytes	0	**0 - 1** - 2 - 3
	1	**1 - 2** - 3 - 0
	2	**2 - 3** - 0 - 1

Table 12-15. Memory Controller–Data Beat Ordering for the 64-Bit Interface

Transfer Size	Starting Double-Word Offset	Double-Word Sequence to/from DRAM and Queues
24 bytes	0	**0 - 1 - 2** - 3
	1	**1 - 2 - 3** - 0
32 bytes	0	0 - 1 - 2 - 3
	1	1 - 2 - 3 - 0
	2	2 - 3 - 0 - 1
	3	3 - 0 - 1 - 2

Note: All underlined word offsets are valid for the transaction.

12.4.7 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode that allows an open page for each logical bank of DRAM. In closed page mode for DDR SDRAMs, the DDR memory controller uses the auto-precharge feature, which allows the controller to indicate the DDR SDRAM that it must automatically close the page after the read or write access. This is performed by using MA10 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. However, it can be separately enabled or disabled for each chip select. In open page mode, the DDR memory controller 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 CSx_CONFIG[AP_x_EN].

12.5 Error Checking and Correction

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core Initiator and system memory. The DDR memory controller detects all double-bit errors, detects all multi-bit errors within a nibble, and it corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, its value compared to the single-bit error threshold register. An error is reported when the counter value is equal to the threshold register

value. The single-bit error registers can be programmed so that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes smaller than 64 bits, the DDR memory controller performs a 64-bit 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 64-bit of merged data. 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 encoding for the ECC code is shown in **Table 12-16** and **Table 12-17**. In 32-bit mode, **Table 12-16** is split into 2 halves. The first half, consisting of rows 0–31, is used to calculate the ECC bits for the first 32 data bits of any 64-bit granule of data. This always applies to the odd data beats on the DDR data bus. The second half of the table, consisting of rows 32–63, is used to calculate the ECC bits for the second 32 bits of any 64-bit granule of data. This always applies to the even data beats on the DDR data bus.

Table 12-16. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		
8	•	•					•	
9	•		•				•	
10	•			•			•	
11	•				•		•	
12	•	•				•	•	•
13	•		•			•	•	•
14	•			•		•	•	•
15	•				•	•	•	•
16		•	•					•
17		•		•				•
18		•			•			•
19	•	•			•			
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•
40			•	•			•	
41			•		•		•	
42	•		•		•		•	•
43		•	•		•		•	•
44			•	•		•	•	•
45			•		•	•	•	•
46	•		•		•	•	•	
47		•	•		•	•	•	
48		•				•	•	
49			•			•	•	
50				•		•	•	
51	•					•	•	

Table 12-16. 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
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 12-17. 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								•

12.6 Error Management

The DDR memory controller detects single-bit, multi-bit, memory select, training errors and address/command parity errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in **12.8.40, *DDR SDRAM Memory Error Detect Register (MnERR_DETECT)*** and **12.8.41, *DDR SDRAM Memory Error Disable Register (MnERR_DISABLE)***.

Single-bit errors are counted and reported based on the ERR_SBE register value (see **Table 12-66** on page 12-115). When a single-bit error is detected, the DDR memory controller does the following:

1. Corrects the data.
2. Increments the single-bit error counter ERR_SBE[SBEC].
3. Generates a critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET].
4. Completes the transaction normally.

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the critical interrupt (if enabled, as described in **Table 12-62** on page 12-111).

The DDR memory controller also detects a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in **Table 12-61** on page 12-110). 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. **Table 12-18** describes the errors.

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.

The training 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 12-18. 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 critical interrupt if enabled.	The error control register logs only 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.		
Training	Calibration error	One of the calibration processes executed during the initialization failed		
Parity	Address and Command error	The memory device detects that the parity bit calculated by the controller doesn't correspond to the parity calculated by the memory device.		

12.7 Set-Up and Initialization

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. The DDR memory controller uses this bank map to assert the appropriate $\overline{\text{MCSx}}$ signal for memory accesses according to the provided bank depths. System software also configures the DDR memory controller at system start-up to multiplex the row and column address bits for each bank (see **Table 12-22** on page 12-47). Address multiplexing occurs according to these configuration bits. At system power-up, initialization software (boot code, for example) must set up the programmable parameters in the memory interface configuration registers listed in **Table 12-19**.

Table 12-19. Memory Interface Configuration Register Initialization Parameters

Register	Parameter Bits	Page
Chip-Select Memory Bounds Register (MCSx_BNDS)	Starting Address for Chip Select x (SA*x*) Ending Address for Chip Select x (EA*x*)	**Table 12-21** on page 12-46
Chip-Select Configuration Register (MCSx_CONFIG)	Chip Select x Enable (CS_x_EN) Chip Select x Auto-Precharge Enable (AP_x_EN) ODT for Reads (ODT_RD_CFG) ODT for Writes (ODT_WR_CFG) Number of Bank Bits (BA_BITS_CS_x) Number of Row Bits (ROW_BITS_CS_x) Number of Column Bits (COL_BITS_CS_x) (INTLV_EN_CTL)	**Table 12-22** on page 12-47
Chip Select Configuration 2(MCSx_CONFIG_2)	PASR_CFG	**Table 12-23** on page 12-49
Extended Timing Parameters for Fields in TIMING_CFG_3 (TIMING_CFG_3)	EXT_REFREC EXT_ACTTOPRE EXT_CASLAT CNTL_ADJ	**Table 12-23** on page 12-49 **Table 12-24** on page 12-50

Table 12-19. Memory Interface Configuration Register Initialization Parameters (Continued)

Register	Parameter Bits	Page
Timing Configuration 0 Register (TIMING_CFG_0)	Read-to-Write Turn-Around (RWT) Write-to-Read Turn-Around (WRT) Read-to-Read Turn-Around (RRT) Write-to-Write Turn-Around (WWT) Active Power-Down Exit Timing (ACT_PD_EXIT) Precharge Power-Down Exit Timing (PRE_PD_EXIT) ODT Power-Down Exit Timing (ODT_PD_EXIT) Mode Register Set Cycle Time (MRS_CYC)	**Table 12-25** on page 12-52
Timing Configuration 1 Register (TIMING_CFG_1)	Precharge-to-Activate Interval (PRETOACT) Activate-to-Precharge Interval (ACTTOPRE) Activate to Read/Write Interval for SDRAM (ACTTORW) $\overline{\text{MCAS}}$ Latency from Read Command (CASLAT) Refresh Recovery Time (REFREC) Last Data to Precharge Minimum Interval (WRREC) Activate-to-Activate Interval (ACTTOACT) Last Write Data Pair to Read Interval (WR_TO_RD)	**Table 12-26** on page 12-56
Timing Configuration 2 Register (TIMING_CFG_2)	Additive Latency (ADD_LAT) $\overline{\text{MCAS}}$-to-Preamble Override (CPO) Write Latency (WR_LAT) Read-to-Precharge (RD_TO_PRE) Write Data Delay (WR_DATA_DELAY) Minimum CKE Pulse Width (CKE_PLS) Window for Four Activates (FOUR_ACT)	**Table 12-27** on page 12-60
DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)	Self Refresh Enable (SREN) ECC Enable (ECC_EN) Registereed DIMM Enable (RD_EN) SDRAM Type (SDRAM_Type) Dynamic Power Management Mode (DYN_PWR) 32-Bit Bus Enable (32_BE) 8-Beat Burst Enable (8_BE) Non-Current Auto Precharge (NCAP) 3T Timing Enable (3T_EN) 2T Timing Enable (2T_EN) Half-Strength Drive Enable (HSE) Bypass Initialization (BI)	**Table 12-28** on page 12-65
DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)	Force Self Refresh (FRC_SR) DLL Reset Disable (DLL_RST_DIS) DQS Configuration (DQS_CFG) ODT Configuration (ODT_CFG) Number of Posted Refreshes (NUM_PR) DRAM Data Initialization (D_INIT)	**Table 12-29** on page 12-68
DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)	Extended SDRAM Mode (ESDMODE) SDRAM Mode (SDMODE)	**Table 12-30** on page 12-70
DDR SDRAM Mode Configuration 2 Register (DDR_SDRAM_MODE_2)	Extended SDRAM Mode 2 (ESDMODE2) Extended SDRAM Mode 3 (ESDMODE3)	**Table 12-31** on page 12-71
DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)	Refresh Interval (REFINT) Precharge Interval (BSTOPRE)	**Table 12-34** on page 12-74

Table 12-19. Memory Interface Configuration Register Initialization Parameters (Continued)

Register	Parameter Bits	Page
DDR SDRAM Data Initialization Register (DDR_DATA_INIT)	Initialization Value (INIT_VALUE)	**Table 12-35** on page 12-75
DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)	Clock Adjust (CLK_ADJUST)	**Table 12-36** on page 12-76
DDR SDRAM Initialization Address Register (DDR_INIT_ADDRESS)	Initialization Address (INIT_ADDR)	**Table 12-37** on page 12-77
DDR Initialization Enable (MDDR_INIT_EN)	Use Initialization Address (UIA)	**Table 12-38** on page 12-77
DDR SDRAM Timing Configuration 4 Register (TIMING_CFG_4)	Read-to-Write Turnaround for Same Chip Select (RWT) Write-to-Read Turnaround for Same Chip Select (WRT) Read-to-Read Turnaround for Same Chip Select (RRT) Write-to-Write Turnaround for Same Chip Select (WWT) DDR SDRAM DLL Lock Time (DLL_LOCK)	**Table 12-39** on page 12-78
DDR SDRAM Timing Configuration 5 Register (TIMING_CFG_5)	Read-to-ODT On (RODT_ON) Read-to-ODT Off (RODT_OFF) Write-to-ODT On (WODT_ON) Write-to-ODT Off (WODT_OFF)	**Table 12-40** on page 12-80
DDR ZQ Calibration Control (MDDR_ZQ_CNTL	ZQ Calibration Enable (ZQ_EN) Power-on Reset ZQ Calibration Time (ZQINIT) Normal Operation Full Calibration Time (ZQOPER) Normal Operation Short Calibration Time (ZQCS)	**Table 12-41** on page 12-83
DDR Write Leveling Control (MDDR_WRLVL_CNTL)	Write Leveling Enable (WRLVL_EN) First DQS Pulse Rising Edge after Margining Mode Is Programmed (WRLVL_MRD) ODT Delay after Margining Mode Is Programmed (WRLVL_ODTEN) DQS/$\overline{\text{DQS}}$ Delay after Margining Mode Is Programmed (WRLVL_DQSEN) Write Leveling Sample Time (WRLVL_SMPL) Write Leveling Repetition Time (WRLVL_WLR) Write Leveling Start Time (WRLVL_START)	**Table 12-42** on page 12-84
DDR Write Leveling Control 2 (MDDR_WRLVL_CNTL_2)	Write Leveling Start Time for DQS1 (WRLVL_START_1) Write Leveling Start Time for DQS2 (WRLVL_START_2) Write Leveling Start Time for DQS3 (WRLVL_START_3) Write Leveling Start Time for DQS4 (WRLVL_START_4)	**Table 12-43** on page 12-87
DDR Write Leveling Control 3 (MDDR_WRLVL_CNTL_3)	Write Leveling Start Time for DQS5 (WRLVL_START_5) Write Leveling Start Time for DQS6 (WRLVL_START_6) Write Leveling Start Time for DQS7 (WRLVL_START_7) Write Leveling Start Time for DQS8 (WRLVL_START_8)	**Table 12-44** on page 12-90
DDR Self Refresh Counter (MDDR_SR_CNTR)	Self Refresh Idle Threshold (SR_IT)	**Table 12-46** on page 12-96

Table 12-19. Memory Interface Configuration Register Initialization Parameters (Continued)

Register	Parameter Bits	Page
DDR SDRAM Register Control Words 1 (MDDR_SDRAM_RCW_1)	Register Control Word 0 (RCW0) Register Control Word 1 (RCW1) Register Control Word 2 (RCW2) Register Control Word 3 (RCW3) Register Control Word 4 (RCW4) Register Control Word 5 (RCW5) Register Control Word 6 (RCW6) Register Control Word 7 (RCW7)	**Table 12-47** on page 12-97
DDR SDRAM Register Control Words 2 (MDDR_SDRAM_RCW_2)	Register Control Word 8 (RCW8) Register Control Word 9 (RCW9) Register Control Word 10 (RCW10) Register Control Word 11 (RCW11) Register Control Word 12 (RCW12) Register Control Word 13 (RCW13) Register Control Word 14 (RCW14) Register Control Word 15 (RCW15)	**Table 12-48** on page 12-98
DDR Control Driver Register 1 (MDDRCDR_1)	DDR Driver Hardware Compensation Enable (DHC_EN) Driver Software Override Enable for MDIC (DSO_MDIC_EN) Driver Software Override valeu for MDIC P-Impedance (DSO_MDICPZ) Driver Software Override valeu for MDIC N-Impedance (DSO_MDIC_NZ) Driver Software Override Output Enable for P-Impedance (DSO_MDIC_PZ_OE) Driver Software Override Output Enable for N-Impedance (DSO_MDIC_NZ_OE) ODT Termination Value for I/O (ODT) Driver Software Override Enable for Address/Command (DSO_C_EN) Driver Software Override Enable for Data (DSO_D_EN) DDR Driver Software override value for Command P-Impedance override (DSO_CPZ) DDR Driver Software override value for Command N-Impedance override (DSO_CNZ) DDR Driver Software override value for Data P-Impedance override (DSO_DPZ) DDR Driver Software override value for Data N-Impedance override (DSO_DNZ)	**Table 12-51** on page 12-101
DDR Control Driver Register 2 (MDDRCDR_2)	Driver Software Override Enable for Clocks (DSO_CLK_EN) DDR Driver Software override value for Clocks P-Impedance override (DSO_CLKPZ) DDR Driver Software override value for Clocks N-Impedance override (DSO_CLKNZ) ODT Termination Value for I/O (ODT)	**Table 12-52** on page 12-104

12.7.1 Programming Differences Between Memory Types

Several fields in the configuration registers must be programmed to reflect the characteristics of the DDR memory used in the application. **Table 12-20** lists several of the characteristics. Refer to the DDR memory specifications to determine the required settings. **Table 12-20** does not list all fields that must be programmed.

Table 12-20. Programming Differences Between Memory Types

Parameter	Description	Differences	
AP*n*_EN	Chip Select *n* Auto Precharge Enable	DDR2	Can be used to place chip select *n* in auto precharge mode
		DDR3	Can be used to place chip select *n* in auto precharge mode
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select typically does not use ODT when issuing reads to the memory.
		DDR3	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select typically does not use ODT when issuing reads to the memory.
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT is typically set to assert for the chip select that is getting written to (value would be set to 001).
		DDR3	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT typically is set to assert for the chip select that is getting written to (value would be set to 001).
ODT_PD_EXIT	ODT Powerdown Exit	DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{AXPD}.
		DDR3	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	DDR2	Should be set according to the specifications for the memory used (t_{RP})
		DDR3	Should be set according to the specifications for the memory used (t_{RP})
ACTTOPRE	Activate to Precharge Timing	DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})
		DDR3	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	DDR2	Should be set according to the specifications for the memory used (t_{RCD})
		DDR3	Should be set according to the specifications for the memory used (t_{RCD})

Table 12-20. Programming Differences Between Memory Types (Continued)

Parameter	Description	Differences	
CASLAT	CAS Latency	DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency
		DDR3	Should be set, along with the Extended CAS Latency, to the desired CAS latency
REFREC	Refresh Recovery	DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})
		DDR3	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})
WRREC	Write Recovery	DDR2	Should be set according to the specifications for the memory used (t_{WR})
		DDR3	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*	DDR2	Should be set according to the specifications for the memory used (t_{RRD})
		DDR3	Should be set according to the specifications for the memory used ($t_{RRD)}$
WRTORD	Write to Read Timing	DDR2	Should be set according to the specifications for the memory used (t_{WTR})
		DDR3	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	DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]
		DDR3	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	DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency if 5 cycles, then this field should be set to 100 (4 cycles).
		DDR3	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.
RD_TO_PRE	Read to Precharge Timing	DDR2	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.
		DDR3	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	DDR2	Should be set according to the specifications for the memory used (t_{CKE})
		DDR3	Should be set according to the specifications for the memory used (t_{CKE})

Table 12-20. Programming Differences Between Memory Types (Continued)

Parameter	Description	Differences	
FOUR_ACT	Four Activate Window	DDR2	Should be set according to the specifications for the memory used (t_{FAW}). Only applies to eight logical banks.
		DDR3	Should be set according to the specifications for the memory used (t_{FAW}).
RD_EN	Registered DIMM Enable	DDR2	If registered are used, then this field should be set to 1
		DDR3	If registered are used, then this field should be set to 1
8_BE	8-beat burst enable	DDR2	Should be set to 0
		DDR3	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 is needed to ensure t_{CCD} is met.
2T_EN	2T Timing Enable	DDR2	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.
		DDR3	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.
DLL_RST_DIS	DLL Reset Disable	DDR2	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.
		DDR3	Should be set to 1
DQS_CFG	DQS Configuration	DDR2	Should be set to 01
		DDR3	Should be set to 01
ODT_CF	ODT Configuration	DDR2	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.
		DDR3	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	DDR2	Should be set to 0
		DDR3	Can be set to 1 if on-the-fly burst chop is 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)	DDR2	Should typically be set to 0000
		DDR3	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.
WRT	Write-to-read turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000
		DDR3	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] is 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)	DDR2	Should typically be set to 0000
		DDR3	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).

Table 12-20. Programming Differences Between Memory Types (Continued)

Parameter	Description	Differences	
WWT	Write-to-write turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000
		DDR3	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).
ZQ_EN	ZQ Calibration Enable	DDR2	Should be set to 0
		DDR3	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	DDR2	Should be set to 0
		DDR3	Can be set to 1 if write leveling is desired. Otherwise the value used in TIMING_CFG_2[WR_DATA_DELAY] is used to shift all bytes during writes to DRAM. If write leveling is used, all other fields in DDR_WRLVL_CNTL should be programmed appropriately based on the DRAM specifications.
BSTOPRE	Burst To Precharge Interval	DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.
		DDR3	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.

12.7.2 DDR SDRAM Initialization Sequence

After all parameters are configured, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. You must allow 200 µs (500 µs for DDR3) to elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before setting MEMEN. Therefore, a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] bit is not set to bypass initialization, the DDR memory controller conducts an automatic initialization sequence to the memory, which follows the memory specifications. If the bypass initialization mode is used, software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

12.8 Memory Controller Programming Model

In the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- Read/write, read only, and write only (R/W, R, and W, respectively) indicate that all the non-reserved fields in a register have the same access type.
- Non-reserved fields that are cleared by writing 1s to them are indicated by w1c.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. For special access registers, read the figure and field descriptions very carefully.

The DDR memory controller registers are as follows:

- Chip-Select Memory Bounds Register (MnCSx_BNDS), **page 12-46**.
- Chip-Select Configuration Register (MnCSx_CONFIG), **page 12-47**.
- Chip-Select Configuration Register 2 (MnCSx_CONFIG_2), **page 12-49**
- DDR SDRAM Timing Configuration 3 (MnTIMING_CFG_3), **page 12-50**.
- DDR SDRAM Timing Configuration 0 Register (MnTIMING_CFG_0), **page 12-52**.
- DDR SDRAM Timing Configuration 1 Register (MnTIMING_CFG_1), **page 12-55**.
- DDR SDRAM Timing Configuration 2 Register (MnTIMING_CFG_2), **page 12-60**.
- DDR SDRAM Control Configuration Register (MnDDR_SDRAM_CFG), **page 12-65**.
- DDR SDRAM Control Configuration 2 Register (MnDDR_SDRAM_CFG_2), **page 12-67**.
- DDR SDRAM Mode Configuration Register (MnDDR_SDRAM_MODE), **page 12-70**.
- DDR SDRAM Mode Configuration 2 Register (MnDDR_SDRAM_MODE_2), **page 12-71**.
- DDR SDRAM Mode Control Register (MnDDR_SDRAM_MD_CNTL), **page 12-71**.

- DDR SDRAM Interval Configuration Register (MnDDR_SDRAM_INTERVAL), **page 12-74**.
- DDR SDRAM Data Initialization Register (MnDDR_DATA_INIT), **page 12-75**.
- DDR SDRAM Clock Control Configuration Register (MnDDR_SDRAM_CLK_CNTL), **page 12-75**.
- DDR Initialization Address Register (MnDDR_INIT_ADDRESS), **page 12-76**.
- DDR Initialization Enable Extended Address (MnDDR_INIT_ENXT_ADDR), **page 12-77**
- DDR SDRAM Timing Configuration 4 (MnTIMING_CFG_4), **page 12-78**
- DDR SDRAM Timing Configuration 5 (MnTIMING_CFG_5), **page 12-80**
- DDR ZQ Calibration Control (MnDDR_ZQ_CNTL), **page 12-82**
- DDR Write Leveling Control (MnDDR_WRLVL_CNTL), **page 12-84**
- DDR Write Leveling Control 2 (MnDDR_WRLVL_CNTL_2), **page 12-87**
- DDR Write Leveling Control 3 (MnDDR_WRLVL_CNTL_3), **page 12-90**
- DDR Pre-Drive Conditioning Control (MnDDR_PD_CNTL), **page 12-93**
- DDR SDRAM Self Refresh Counter (MnDDR_SR_CNTR), **page 12-96**
- DDR SDRAM Register Control Words 1 (MnDDR_SDRAM_RCW_1), **page 12-97**
- DDR SDRAM Register Control Words 2 (MnDDR_SDRAM_RCW_2), **page 12-98**
- DDR Debug Status Register 1 (MnDDRDSR_1), **page 12-99**
- DDR Debug Status Register 2 (MnDDRDSR_2), **page 12-100**
- DDR Control Driver Register 1 (MnDDRCDR_1), **page 12-100**
- DDR Control Driver Register 2 (MnDDRCDR_2), **page 12-104**
- DDR IP Block Revision 1 Register (MnDDR_IP_REV1), **page 12-105**.
- DDR IP Block Revision 2 Register (MnDDR_IP_REV2), **page 12-105**.
- Memory Data Path Error Injection Mask High Register (MnDDR_ERR_INJECT_HI), **page 12-106**.
- Memory Data Path Error Injection Mask Low Register (MnDDR_ERR_INJECT_LO), **page 12-106**.
- Memory Data Path Error Injection Mask ECC Register (MnDDR_ERR_INJECT), **page 12-107**
- Memory Data Path Read Capture High Register (MnCAPTURE_DATA_HI), **page 12-108**.
- Memory Data Path Read Capture Low Register (MnCAPTURE_DATA_LO), **page 12-108**.
- Memory Data Path Read Capture ECC Register (MnCAPTURE_ECC), **page 12-109**.
- Memory Error Detect Register (MnERR_DETECT), **page 12-109**.
- Memory Error Disable Register (MnERR_DISABLE), **page 12-110**.
- Memory Error Interrupt Enable Register (MnERR_INT_EN), **page 12-112**.
- Memory Error Attributes Capture Register (MnCAPTURE_ATTRIBUTES), **page 12-113**.
- Memory Error Address Capture Register (MnCAPTURE_ADDRESS), **page 12-114**.

- Single-Bit ECC Memory Error Management Register (MnERR_SBE), **page 12-114**.
- Debug Register 2 (MnDEBUG_2), **page 12-115**

Note: DDR1 controller uses base address: 0xFFF20000. DDR2 controller uses base address: 0xFFF22000.

12.8.1 Chip-Select Bounds (MnCSx_BNDS)

CS0_BNDS Chip-Select Bounds Register Offset 0x0000
CS1_BNDS 0x0008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								SAx							
Type	R								R/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**
	—								EAx							
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSx_BNDS defines the starting and ending address of the memory space that corresponds to the individual chip selects.

Note: The size specified in CSx_BNDS should equal the size of physical DRAM. Also, note that EA*x* must be greater than or equal to SA*x*. The 8 msb of the address is used to define the SAx and EAx.

If chip select interleaving is enabled, all fields in the lower interleaved chip select are used, and the other chip selects bounds registers are not used. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_BNDS are used, and all fields in CS1_BNDS are not used.

Table 12-21. CSx_BNDS Field Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Cleared to zero for future compatibility.
SAx 23–16	0	**Starting Address** Specifies the starting address for chip-select (bank) *x*. This value is compared against the 8 MSBs of the 32-bit address. See the previous note.
— 15–8	0	Reserved. Cleared to zero for future compatibility.
EA*x* 7–0	*0*	**Ending Address** Specifies the ending address for chip select (bank) *x*. This value is compared against the 8 MSBs of the 32-bit address. See the previous note.

12.8.2 Chip-Select x Configuration Register (MnCSx_CONFIG)

CS0_CONFIG Chip-Select 0 Configuration Register Offset 0x0080
CS1_CONFIG Chip-Select 1 Configuration Register Offset 0x0084

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CS_x_EN	—							AP_x_EN	ODT_RD_CFG			—	ODT_WR_CFG		
Type	R/W	R/W		R	R/W								R	R/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**
	BA_BITS_CS_x	—			ROW_BITS_CS_x				—					COL_BITS_CS_x		
Type	R/W	R			R/W				R					R/W		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CSx_CONFIG register enables the DDR chip select x and sets the number of row and column bits used for the chip select. The register should be loaded with the correct number of row and column bits for each SDRAM. Because the ROW_BITS_CS_x and COL_BITS_CS_x fields establish address multiplexing, it is essential to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select are used, and the other registers' fields are 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 are used, but only the ODT_RD_CFG and ODT_WR_CFG fields in CS1_CONFIG are used.

Table 12-22. CSx_CONFIG Field Descriptions

Bit	Reset	Description	Settings
CS_x_EN 31	0	**Chip Select *x* Enable** Enables/disables chip select.	0 Chip select x is not active. 1 Chip select x is active and assumes the state set in CSx_BNDS.
— 30-24	0	Reserved. Write to zero for future compatibility.	
AP_x_EN 23	0	**Chip Select x Auto-Precharge Enable** Specifies when auto-precharged is enabled for chip select x.	0 Chip select x is auto-precharged only if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select x always issues an auto-precharge for read and write transactions.
ODT_RD_CFG 22–20	0	**On-Die Termination (ODT) for Reads** Specifies when ODT is to be asserted for read accesses. Note that $\overline{\text{CAS}}$ latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled.	000 Never assert ODT for reads. 001 Assert ODT only during reads to CSx. 010 Assert ODT only during reads to other chip selects. 011 Reserved. 100 Assert ODT for all reads. 101–111 Reserved.

Table 12-22. CSx_CONFIG Field Descriptions (Continued)

Bit	Reset	Description	Settings
— 19	0	Reserved. Write to zero for future compatibility.	
ODT_WR_CFG 18–16	0	**ODT for Writes** Specifies when ODT is to be asserted for write accesses. Note that write latency plus additive latency must be at least 3 cycles for ODT_WR_CFG to be enabled.	000 Never assert ODT for writes. 001 Assert ODT only during writes to CSx. 010 Assert ODT only during writes to other chip selects. 011 Reserved. 100 Assert ODT for all writes. 101–111 Reserved.
BA_BITS_CS_x 15–14	0	**Number of Bank Bits** Specifies the number of logical bank bits MBA[2–0] for SDRAM on chip select x. See **Table 12-6** and **Table 12-7** for details	00 2 logical bank bits. 01 3 logical bank bits. 10–11 Reserved
— 13–11	0	Reserved. Write to zero for future compatibility.	
ROW_BITS_CS_x 10–8	0	**Number of Row Bits** Specifies the number of row bits for SDRAM on chip select x. See **Table 12-6** and **Table 12-7** for details	000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 100 16 row bits 101–111 Reserved
— 7–3	0	Reserved. Write to zero for future compatibility.	
COL_BITS_CS_x 2–0	0	**Number of Column Bits** Specifies the number of column bits for SDRAM on chip select x. See **Table 12-6** and **Table 12-7** for details.	000 8 column bits. 001 9 column bits. 010 10 column bits. 011 11 column bits. 100–111 Reserved.

12.8.3 Chip-Select x Configuration Register 2 (MnCSx_CONFIG_2)

CS0_CONFIG_2 Chip-Select x Configuration Register 2 Offset 0x00C0
CS1_CONFIG_2 0x00C4

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

CSx_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 are used, and the other register fields are unused..

Table 12-23. CSx_CONFIG Field Descriptions

Bit	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	
PASR_CFG 26–24	0	**Partial Array Self Refresh Configuration** Controls the bits that is placed on MA[2:0] during the write to the EMRS(2) register DDR2/DDR3 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 overrides the least significant 3 bits in DDR_SDRAM_MODE_2[ESDMODE2] for DDR2/DDR3 during the automatic initialization for chip select *x*. In addition, if a non-zero value is programmed in this field, then the address decode for chip select *x* is optimized for partial array self refresh (see **Section 12.2.3**),	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 EMRS2 is only supported for DDR2 and DDR3 memory types.
— 23–0	0	Reserved. Write to zero for future compatibility.	

12.8.4 DDR SDRAM Timing Configuration 3 (MnTIMING_CFG_3)

TIMING_CFG_3 DDR SDRAM Timing Configuration 3 Register Offset 0x0100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—							EXT_ ACTTO PRE	—				EXT_REFREC			
Type	R							R/W	R				R/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**
	—			EXT_ CAS LAT	—									CNTL_ADJ		
Type	R			R/W	R									R/W		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_3 sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

Table 12-24. TIMING_CFG_3 Bit Descriptions

Bits	Reset	Description	Setting
— 31–25	0	Reserved. Write to zero for future compatibility.	
EXT_ACTTOPRE 24	0	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 TIMING_CFG_1[ACTTOPRE] to obtain a 5-bit value for the total activate to precharge. Note that a 5-bit value of 0_0000 is the same as a 5-bit value of 1_0000. Both values represent 16 cycles.	0 0 clocks 1 16 clocks
— 23–20	0	Reserved. Write to zero for future compatibility.	
EXT_ REFREC 19–16	0	**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 combined with TIMING_CFG_1[REFREC] to obtain the total refresh recovery time. Note that the minimum value for REFREC is 8 clock cycles. t_{RFC} = {REFREC | + | EXT_REFREC} min. value = 8 clocks (REFREC = 0x0) + EXT_REFREC = 0x0 max. value = 240+ 23 = 263	0000 0 clock cycles. 0001 16 clock cycles. 0010 32 clock cycles. 0011 48 clock cycles. 0100 64 clock cycles. 0101 80 clock cycles. 0110 96 clock cycles. 0111 112 clock cycles 1000 128 clock cycles 1001 144 clock cycles 1010 160 clock cycles 1011 176 clock cycles 1100 192 clock cycles 1101 208 clock cycles 1110 224 clock cycles 1111 240 clock cycles

Table 12-24. TIMING_CFG_3 Bit Descriptions (Continued)

Bits	Reset	Description	Setting
— 15–13	0	Reserved. Write to zero for future compatibility.	
EXT_CASLAT 12	0	Extended $\overline{\text{MCAS}}$ 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 if this bit is set, then 8 clocks are added to the programmed value in TIMING_CFG_1[CASLAT].	0 0 clocks 1 8 clocks
— 11–3	0	Reserved. Write to zero for future compatibility.	
CNTL_ADJ 2-0	0	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[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1]	000 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] is launched aligned with the other DRAM address and control signals. 001 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] is launched 1/4 platform cycle later than the other DRAM address and control signals. 010 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] is launched 1/2 platform cycle later than the other DRAM address and control signals. 011 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] is launched 3/4 platform cycles later than the other DRAM address and control signals. 100 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] is launched 1platform cycles later than the other DRAM address and control signals. 101 MODT[0:1], $\overline{\text{MCS}}$[0:1], and MCKE[0:1] is launched 5/4 platform cycles later than the other DRAM address and control signals. 110-111Reserved

12.8.5 DDR SDRAM Timing Configuration Register 0 (MnTIMING_CFG_0)

TIMING_CFG_0 DDR SDRAM Timing Configuration Register 0 Offset 0x0104

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RWT		WRT		RRT		WWT		—	ACT_PD_EXIT			PRE_PD_EXIT			
Type	R/W								R	R/W			R/W			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—				ODT_PD_EXIT				—				MRS_CYC			
Type	R				R/W				R				R/W			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1

TIMING_CFG_0 sets the number of clock cycles between various SDRAM control commands.

Table 12-25. TIMING_CFG_0 Field Descriptions

Bit	Reset	Description	Settings
RWT 31–30	0	**Read-to-Write Turn-Around (t_{RTW})** Specifies how many extra cycles to add between a read-to-write turnaround. If 0 clock cycles is chosen, then the DDR controller uses a fixed number based on the $\overline{CAS}$ latency and write latency. A value other than 0 adds extra cycles to this default calculation. By default, the DDR controller determines the read-to-write turn-around as CL – WL + BL/2 + 2. CL is the $\overline{CAS}$ latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length.	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.
WRT 29–28	0	**Write-to-Read Turn-Around** Specifies how many extra cycles to add between a write-to-read turn-around. If 0 clock cycles is chosen, then the DDR controller uses a fixed number based on the read latency and write latency. A value other than 0 adds extra cycles to this default calculation. By default, the DDR controller determines the write-to-read turn-around as WL – CL + BL/2 + 1. CL is the $\overline{CAS}$ latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length.	00 0 clock cycles. 01 1 clock cycle. 10 2 clock cycles. 11 3 clock cycles.
RRT 27–26	0	**Read-to-Read Turn-Around** Specifies how many extra cycles to add between reads to different chip selects. By default, 3 cycles are required between read commands to different chip selects. If 00 is selected the DDR Controller uses a predefined value - 3 clocks for the turnaround, selecting a value other than 00 adds extra cycles to this predefined value according to the selection When DDR works in 8 beat burst the default is 5 clock cycles. Note: DDR2 does not support 8-beat bursts.	*DDR2:* 00 3 clock cycles. 01 4 clock cycle. 10 5 clock cycles. 11 6 clock cycles.
WWT 25–24	0	**Write-to-Write Turn-Around** Specifies how many extra cycles to add between writes to different chip selects. By default, 2 cycles are required between write commands to different chip selects. If 00 is selected the DDR Controller uses a predefined value - 2 clocks for the turnaround, selecting a value other than 00 adds extra cycles to this predefined value according to the selection When DDR works in 8 beat burst the default is 4 clock cycles. Note: DDR2 does not support 8-beat bursts.	*DDR2:* 00 2 clock cycles. 01 3 clock cycle. 10 4 clock cycles. 11 5 clock cycles.
— 23	0	Reserved. Write to zero for future compatibility.	

Table 12-25. TIMING_CFG_0 Field Descriptions (Continued)

Bit	Reset	Description	Settings
ACT_PD_EXIT 22–20	0b001	**Active Power-Down Exit Timing (t_{XARD} and t_{XARDS})** Specifies how many clock cycles to wait between exit from active power-down and issuing a command. The default is one clock cycle.	000 Reserved 001 1 clock cycles. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles
PRE_PD_EXIT 19–16	0b0001	**Precharge Power-Down Exit Timing (t_{xp})** Specifies how many clock cycles to wait after exiting precharge power-down before issuing any command. The default is one clock cycle.	0000 Reserved. 0001 1 clocks. 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles 1000 8 clock cycles 1001 9 clock cycle. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.
— 15–12	0	Reserved. Write to zero for future compatibility.	
ODT_PD_EXIT 11–8	0b0001	**ODT Power-Down Exit Timing (t_{AXPD})** Specifies how many clock cycles must pass after exit from power-down before ODT can be asserted. The default is 1 clock cycle. **Note:** For DDR3, ODT_PD_EXIT must be greater than TIMING_CFG_5[RODT_ON] when using RODT_ON overrides and must be grater than TIMING_CFG_5[WODT_ON] when using WODT_ON overrides.	0000 0 clock 0001 1 clock 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles. 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.
— 7–4	0	Reserved. Write to zero for future compatibility.	

Table 12-25. TIMING_CFG_0 Field Descriptions (Continued)

Bit	Reset	Description	Settings
MRS_CYC 3–0	0b0101	**Mode Register Set Cycle Time (t_{MRD})** Specifies the number of clock cycles that must pass between a Mode Register Set command and another command. The default is 5 clock cycles.	0000 Reserved 0001 1 clock 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles. 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.

12.8.6 DDR SDRAM Timing Configuration Register 1 (MnTIMING_CFG_1)

TIMING_CFG_1 DDR SDRAM Timing Configuration Register 1 Offset 0x0108

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PRETOACT				ACTTOPRE				ACTTORW				CASLAT			
Type	R/W				R/W				R/W				R/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**
	REFREC				WRREC				—	ACTTOACT			—	WRTORD		
Type	R/W				R/W				R	R/W			R	R/W		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_1 sets the number of clock cycles between various SDRAM control commands.

Table 12-26. TIMING_CFG_1 Field Descriptions

Bits	Reset	Description	Settings
PRETOACT 31–28	0	**Precharge-to-Activate Interval (t_{RP})** Specifies the minimum number of clock cycles between a precharge command and an activate or refresh command. This number is calculated from the AC specifications of the SDRAM. This field must be programmed for proper operation of the DDR Controller.	0000 Reserved. 0001 1 clock 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.
ACTTOPRE 27–24	0	**Activate to Precharge Interval (t_{RAS})** Specifies the minimum number of clock cycles between an activate command and a precharge command. This number is calculated from the AC specifications of the SDRAM. This field is concatenated with TIMING_CFG_3[EXT_ACTTOPRE] to obtain a 5-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] = 0, but it is equal to 0-3 clocks when TIMING_CFG_3[EXT_ACTTOPRE] = 1. This field must be programmed for proper operation of the DDR Controller.	0000 16 clock cycles. 0001 17 clock cycles. 0010 18 clock cycles. 0011 19 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles. 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.

Table 12-26. TIMING_CFG_1 Field Descriptions (Continued)

Bits	Reset	Description	Settings
ACTTORW 23–20	0	**Activate to Read/Write Interval for SDRAM (t_{RCD})** Specifies the minimum number of clock cycles between an activate command and a read or write command. This interval is calculated from the AC specifications of the SDRAM. This field must be programmed for proper operation of the DDR Controller.	0000 Reserved. 0001 1 clock cycle. 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.
CASLAT 19–16	0	**$\overline{\text{MCAS}}$ Latency from READ Command** Specifies the number of clock cycles between the time the SDRAM registers a READ command and the availability of the first output data. If a READ command is registered at clock edge *n* and the latency is *m* clock cycles., 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 **Section 12.8.9**, *DDR SDRAM Control Configuration Register 2 (MnDDR_SDRAM_CFG_2)* on **page 12-70**. This field must be programmed for proper operation of the DDR Controller.	0000 Reserved. 0001 1 clock cycle. 0010 Reserved. 0011 2 clock cycles. 0100 Reserved. 0101 3 clock cycles. 0110 Reserved. 0111 4 clock cycles. 1000 Reserved. 1001 5 clock cycles. 1010 Reserved. 1011 6 clock cycles. 1100 Reserved. 1101 7 clock cycles. 1110 Reserved. 1111 8 clock cycles.

Table 12-26. TIMING_CFG_1 Field Descriptions (Continued)

Bits	Reset	Description	Settings
REFREC 15–12	0	**Refresh Recovery Time (t_{RFC})** Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is combined with TIMING_CFG_3[EXT_REFREC] to obtain the total refresh recovery time. Note that the minimum value for REFREC is 8 clock cycles. t_{RFC} = {REFREC || EXT_REFREC} + 8 min. value = 8 clocks (REFREC = 0x0) + EXT_REFREC = 0x0 max. value = 240 + 15 + 8 = 263 This required value can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a minimum refresh to activate interval in nanoseconds.	0000 8 clock cycles. 0001 9 clock cycles. 0010 10 clock cycles. 0011 11 clock cycles. 0100 12 clock cycles. 0101 13 clock cycles. 0110 14 clock cycles. 0111 15 clock cycles. 0000 16 clock cycles. 0001 17 clock cycles. 0010 18 clock cycles. 0011 19 clock cycles. 0100 20 clock cycles. 0101 21 clock cycles. 0110 22 clock cycles. 1111 23 clock cycles.
WRREC 11–8	0	**Write Recovery (t_{WR})** Specifies the minimum number of clock cycles between the last data associated with a write command and a precharge command. This interval, write recovery time, is calculated from the AC specifications of the SDRAM. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to *(t_{WR} + 2* cycles). This field must be programmed for proper operation of the DDR Controller. **Note:** DDR_SDRAM_CFG_2[OBC_CFG] = 1 is the recommended mode.	0000 0 clock cycle. 0001 1 clock cycle. 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles. 0110 6 clock cycles. 0111 7 clock cycles 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.
— 7	0	Reserved. Write to zero for future compatibility.	
ACTTOACT 6–4	0	**Activate-to-Activate Interval (t_{RRD})** Specifies the minimum number of clock cycles between an activate command and another activate command for a different logical bank in the same physical bank (chip select). This interval is calculated from the AC specifications of the SDRAM. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles.

Table 12-26. TIMING_CFG_1 Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 3	0	Reserved. Write to zero for future compatibility.	
WRTORD 2–0	0	**Last Write Data Pair to Read Command Interval (t_{WTR})** Specifies the minimum 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) This field must be programmed for proper operation of the DDR Controller. **Note:** DDR_SDRAM_CFG_2[OBC_CFG] = 1 is the recommended mode.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycles. 110 6 clock cycles. 111 7 clock cycles.

12.8.7 DDR SDRAM Timing Configuration Register 2 (MnTIMING_CFG_2)

TIMING_CFG_2 DDR SDRAM Timing Configuration Register 2 Offset 0x010C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ADD_LAT				CPO					WR_LAT				—		
Type	R/W													R		
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**
	RD_TO_PRE			WR_DATA_ DELAY			—	CKE_PLS			FOUR_ACT					
Type	R/W						R	R/W								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMING_CFG_2 sets the clock delay to data for writes and should be defined according to the system timing. **Table 12-28** describes the TIMING_CFG_2 fields.

Table 12-27. TIMING_CFG_2 Field Descriptions

Bit	Reset	Description	Settings
ADD_LAT 31–28	0	**Additive Latency** Specifies the number of clock cycles from the Posted CAS operation until the registered read/write command is issued inside the SDRAM as defined in the JEDEC DDR2 and DDR3 standards. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. **Note:** For DDR2 ADD_LAT can be 0–4. For DDR3 ADD_LAT can be 0–2.	0000 0 clock cycles. 0001 1 clock cycle. 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles. 0101 5 clock cycles 0110 6 clock cycles 0111 7 clock cycles 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.

Table 12-27. TIMING_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	Settings
CPO 27–23	0	**$\overline{\text{MCAS}}$-to-Preamble Override** Defines the number of DRAM cycles between a read command and the time the corresponding DQS preamble is valid for the memory controller. For these decodings, read latency (RL) is equal to the $\overline{\text{CAS}}$ latency plus the additive latency. For CPO decodings other than 00000 and 11111, read latency is rounded up to the next integer value. In other words, CPO timing decides when the DDR controller starts to wait for the first DQS rising edge from DDR memory during the DQS preamble for read access. The DQS rising edge should occur within 1 clock cycle from the timing defined by this parameter.	00000 RL + 1 00001 Reserved 00010 RL 00011 RL + 1/4 00100 RL + 1/2 00101 RL + 3/4 00110 RL + 1 00111 RL + 5/4 01000 RL + 3/2 01001 RL + 7/4 01010 RL + 2 01011 RL + 9/4 01100 RL + 5/2 01101 RL + 11/4 01110 RL + 3 01111 RL + 13/4 10000 RL + 7/2 10001 RL + 15/4 10010 RL + 4 10011 RL + 17/4 10100 RL + 9/2 10101 RL + 19/4 10110 RL + 5 10111 RL + 21/4 11000 RL + 11/2 11001 RL + 23/4 11010 RL + 6 11011 RL + 25/4 11100 RL + 13/2 11101 RL + 27/4 11110 RL + 7 11111 Automatic Calibration (recommended)

Table 12-27. TIMING_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	Settings
WR_LAT 22–19	0	**Write Latency** Specifies the number of clock cycles between the write command and the first data write when AL = 0. When additive latency is applied, WR_LAT specifies the number of clock cycles from when the SDRAM issues the registered write command to the first data write. This field must be programmed for proper operation of the DDR Controller.	0000 Reserved. 0001 1 clock cycle. 0010 2 clock cycles. 0011 3 clock cycles. 0100 4 clock cycles 0101 5 clock cycles 0110 6 clock cycles 0111 7 clock cycles 1000 8 clock cycles. 1001 9 clock cycles. 1010 10 clock cycles. 1011 11 clock cycles. 1100 12 clock cycles. 1101 13 clock cycles. 1110 14 clock cycles. 1111 15 clock cycles.
— 18–16	0	Reserved. Write to zero for future compatibility.	
RD_TO_PRE 15–13	0	**Read to Precharge (t_{RTP})** When AL = 0 then RD_TO_PRE specifies the number of clock cycles between the read command and the precharge command. When additive latency is applied, RD_TO_PRE specifies minimum tRTP timing from the registered read command issued internally by the SDRAM to the precharge command. The interval between the posted read command and the precharge command is ADD_LAT + RD_TO_PRE cycles. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to (t_{RTP} + *2* cycles) This field must be programmed for proper operation of the DDR Controller. **Note:** DDR_SDRAM_CFG_2[OBC_CFG] = 1 is the recommended mode.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycle. 110 6 clock cycles. 111 7 clock cycles.

Table 12-27. TIMING_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	Settings
WR_ DATA_ DELAY 12–10	0	**Write Data Delay** Specifies the adjust timing of DQS and data from its associated clock edge during write cycles. The adjust timing range step sizes are in 1/4 SDRAM clock periods. The tDQSS specification must be within ±25% of the SDRAM clock in the case of DDR2.This parameter is used to meet the tDQSS timing requirement. Using the same clock delay setting for TIMING_CFG_2[WR_DATA_DELAY] and DDR_SDRAM_CLK_CNTL[CLK_ADJUST] ideally results in no tDQSS skew. The write preamble is typically driven high for 1/2 DRAM cycle, and then it is driven low for 1/2 DRAM cycle. However, for WR_DATA_DELAY settings of 0 clocks and 1/4 clocks, the write preamble is 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. **Figure 12-11** is an example of the timing definition of WR_DATA_DELAY when using CLK_ADJUST with a 1/2 clock delay. **Note:** The recommended value for this field is 0b010.	000 0 clock delay 001 1/4 clock delay 010 1/2 clock delay 011 3/4 clock delay 100 1 clock delay 101 5/4 clock delay 110 3/2 clock delay 111 Reserved
— 9	0	Reserved. Write to zero for future compatibility.	
CKE_PLS 8–6	0	**Minimum CKE Pulse Width (t_{CKE})** Specifies the minimum clock cycles tCKE that SDRAM must remain in Self-Refresh mode. This field must be programmed for proper operation of the DDR Controller.	000 Reserved. 001 1 clock cycle. 010 2 clock cycles. 011 3 clock cycles. 100 4 clock cycles. 101 5 clock cycle. 110 6 clock cycles. 111 7 clock cycles.

Table 12-27. TIMING_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	Settings
FOUR_ACT 5–0	0	**Window for Four Activates (t_{FAW}).** Specifies the tFAW timing. This is applied to DDR2/DDR3 with eight logical banks only. See Section 2.5 Bank Active Command in JESD79-2C, p. 28.	000000 Reserved. 000001 1 clock cycle. 000010 2 clock cycles. 000011 3 clock cycles. 000100 4 clock cycles. 000101 5 clock cycle. 000110 6 clock cycles. 000111 7clock cycles. 001000 8clock cycles. 001001 9 clock cycle. 001010 10 clock cycles. 001011 11 clock cycles. 001100 12 clock cycles. 001101 13 clock cycle. 001110 14 clock cycles. 001111 15 clock cycles. 010000 16 clock cycles. 010001 17 clock cycle. 010010 18 clock cycles. 010011 19clock cycles. 010100 20clock cycles. 010101 21 clock cycle. 010110 22 clock cycles. 010111 23 clock cycles. 011000 24 clock cycles. 011001 25 clock cycle. 011010 26 clock cycles. 011011 27clock cycles. 011100 28clock cycles. 011101 29 clock cycle. 011110 30 clock cycles. 011111 31 clock cycles. 100000 32 clock cycles. 100001–111111 Reserved.

12.8.8 DDR SDRAM Control Configuration Register (MnDDR_SDRAM_CFG)

DDR_SDRAM_CFG DDR SDRAM Control Configuration Register Offset 0x0110

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MEM_EN	SREN	ECC_EN	RD_EN	—	SDRAM_TYPE			—		DYN_PWR_MGMT	—	32_BE	8_BE	NCAP	3T_EN
Type	R/W				R	R/W			R		R/W	R	R/W			
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	2T_EN	BA_INTLV_CTL							—				HSE	—	MEM_HALT	BI
Type	R/W								R				R/W	R	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CFG enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting and dynamic power management.

Table 12-28. DDR_SDRAM_CFG Field Descriptions

Bits	Reset	Description	Settings
MEM_EN 31	0	**DDR SDRAM Interface Logic Enable** Enables/disables SDRAM interface logic. This bit must not be set until the initialization code has appropriately configured all other memory configuration parameters.	0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled.
SREN 30	0	**Self Refresh Enable (during sleep)** Enables/disables self refresh during sleep. When self refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep.	0 SDRAM self refresh is disabled during sleep. 1 SDRAM self refresh is enabled during sleep
ECC_EN 29	0	**ECC Enable** Enables/disables ECC protection mechanism including error reporting and interrupts.	0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
RD_EN 28	0	**Registered DIMM Enable** Specifies the type of DIMM used in the system. **Note:** RD_EN and 2T_EN must not be both set at the same time.	0 Unbuffered DIMM.or discrete memory devices. 1 Registered DIMM.
— 27	0	Reserved. Write to zero for future compatibility.	
SDRAM_ TYPE 26–24	011	**Type of SDRAM Device** Specifies the type of device. The default value is 0b011 to designate DDR2 SDRAM.	011 DDR2 SDRAM. 111 DDR3 SDRAM. All other values reserved.
— 23–22	0	Reserved. Write to zero for future compatibility.	
DYN_PWR 21	0	**Dynamic Power Management Mode** Enabled/disables dynamic power management mode. When this bit is set and there is no on-going memory activity, the SDRAM CKE signal is deasserted.	0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled.

Table 12-28. DDR_SDRAM_CFG Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 20	0	Reserved. Write to zero for future compatibility.	
32_BE 19	0	**32-Bit Bus Enable** Selects bus size.	0 64-bit bus is used. 1 32-bit bus is used.
8_BE 18	0	**8-Beat Burst Enable** **Note:** DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode; DDR3 (SDRAM_TYPE = 111) must use 8-beat bursts when using 32-bit bus mode. DDR3 must use 4 beat burst when using 64-bit bus mode.	0 4-beat burst. 1 8-beat burst.
NCAP 17	0	**Non-Concurrent Auto-Precharge** Some older DDR DRAMs do not support concurrent auto precharge. If one of these devices is used, this bit must be set if auto precharge is used.	0 DRAMs in system support concurrent auto-precharge. 1 DRAMs in system do not support concurrent auto-precharge.
3T_EN 16	0	**3T Timing Enable** Enables/disabled 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. When this bit is cleared, the DRAM command/address are held for only one cycle on the DRAM bus. When this bit is set, the DRAM command/address are held for three full clock cycles. on the DRAM bus for every DRAM transaction. However, the chip select is held only for the third cycle.	0 1T timing is enabled if 2T_EN is also cleared. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 3T timing is enabled.
2T_EN 15	0	**2T Timing Enable** Enables/disabled 2T timing. This field cannot be used with a 32-bit bus if 4-beat bursts are used. When this bit is cleared, the DRAM command/address are held for only one cycle on the DRAM bus. When this bit is set, the DRAM command/address are held for two full clock cycles. on the DRAM bus for every DRAM transaction. However, the chip select is held only for the second cycle.	0 1T timing is used if 3T_EN is also cleared. 1 2T timing is enabled.
BA_INTLV_CTL 14–8	0	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving.	0000000 No external memory banks are interleaved 1000000 External memory banks 0 and 1 are interleaved
— 7–4	0	Reserved. Write to zero for future compatibility.	
HSE 3	0	**Half-Strength Drive Enable** Specifies whether the I/O drivers are configured to full strength or half strength. This bit applies only when automatic driver compensation is disabled and the software override for the driver strength is not used in DDRCDR1 and 2.	0 I/O drivers are configured to full-strength. 1 IO drivers are configured to half-strength.
— 2	0	Reserved. Write to zero for future compatibility.	

Table 12-28. DDR_SDRAM_CFG Field Descriptions (Continued)

Bits	Reset	Description	Settings
MEM_HALT 1	0	**DDR Memory Controller Halt** When this bit is set, the memory controller does not accept any new transactions until the bit is cleared. This bit can be used when initialization is bypassed and the MODE REGISTER SET, EXTENDED MODE REGISTER SET, EXTENDED MODE REGISTER2 SET and EXTENDED MODE REGISTER3 SET commands are forced through software. This bit should be used carefully. Using this MHALT option can create congestion on the system interconnection and can cause hangs of the cores and other initiator.	0 DDR controller accepts new transactions. 1 DDR controller finishes any remaining transactions and then halts until software clears this bit.
BI 0	0	**Bypass Initialization** Specifies the conditions for initialization. When this bit is set, software is responsible for initializing memory through the DDR_SDRAM_MD_CNTL register. If software is initializing memory, the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. **Note:** Note that the DDR controller does 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. For details on avoiding ECC errors in this mode, see the discussion of the DDR SDRAM Initialization Address Register on **page 12-76**.	0 DDR controller cycles through the initialization routine based on the value of SDRAM_Type. 1 Initialization routine is bypassed.

12.8.9 DDR SDRAM Control Configuration Register 2 (MnDDR_SDRAM_CFG_2)

DDR_SDRAM_CFG_2 DDR SDRAM Control Configuration Register 2 Offset 0x0114

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FRC_SR	—	DLL_RST_DIS	—	DQS_CFG		—			ODT_CFG		—				
Type	R/W	R	R/W	R	R/W		R			R/W		R				
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**
	NUM_PR				—					OBC_CFG	AP_EN	D_INIT	—	RCW_EN	—	MD_EB
Type	R/W				R					R/W			R	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CFG_2 provides control configuration for the DDR controller in addition to that provided by DDR_SDRAM_CFG.

Table 12-29. DDR_SDRAM_CFG_2 Field Descriptions

Bit	Reset	Description	
FRC_SR 31	0	**Force Self Refresh**	0 Normal operating mode. 1 Enter 'Self Refresh mode.
— 30	0	Reserved. Write to zero for future compatibility.	
DLL_RST_DIS 29	0	**DLL Reset Disable** The DDR controller typically issues a DLL reset to the DRAMs when it exists self refresh. However, you can disable this function by setting this bit during initialization.	0 DDR controller issues a DLL reset when exiting self refresh. 1 DDR controller does not issue a DLL reset when exiting self refresh.
— 28	0	Reserved. Write to zero for future compatibility.	
DQS_CFG 27–26	0	**DQS Configuration** This bit must be programmed for proper operation.	00 Reserved 01 Differential DQS signals are used for DDR2 support. 10 Reserved. 11 Reserved.
— 25–23	0	Reserved. Write to zero for future compatibility.	
ODT_CFG 22–21	0	**ODT Configuration** Defines how ODT is driven to the on-chip I/O. See **Table 12-51** and **Table 12-52** for the definition of the impedance value that is used.	00 Never assert ODT to internal I/O. 01 Assert ODT to internal I/O only during writes to DRAM. 10 Assert ODT to internal I/O only during reads to DRAM. 11 Always keep ODT asserted to internal I/O.
— 20–16	0	Reserved. Write to zero for future compatibility.	
NUM_PR 15–12	0	**Number of Posted Refreshes** Determines how many posted refreshes, if any, can be issued at one time. If posted refreshes are used, this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed so that the maximum t_{ras} specification cannot be violated. For example, some DDR SDRAMs cannot use more than three posted refreshes because the required refresh interval can exceed the maximum constraint for t_{ras}. Note: {TIMING_CFG_1[PRETOACT] + [DDR_SDRAM_CFG_2[NUM_PR] * ({EXT_REFREC || REFREC} + 8 + 2)]} << DDR_SDRAM_INTERVAL[REFINT	0000 Reserved. 0001 1 refresh at a time. 0010 2 refreshes at a time. 0011 3 refreshes at a time. 0100 4 refreshes at a time. 0101 5 refresh at a time. 0110 6 refreshes at a time. 0111 7 refreshes at a time. 1000 8 refreshes at a time. 1001–1111Reserved.
— 11–7	0	Reserved. Write to zero for future compatibility.	

Table 12-29. DDR_SDRAM_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	
OBC_CFG 6	0	**On-The-Fly Burst Chop Configuration** Determines if on-the-fly Burst Chop is 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.	0 On-the-fly Burst Chop mode is disabled. Fixed burst lengths defined in DDR_SDRAM_CFG[8_BE] are used. If fixed Burst Chop is used (with DDR3 memories), then DDR_SDRAM_CFG[8_BE] should be cleared. 1 On-the-fly Burst Chop mode is used. DDR_SDRAM_CFG[8_BE] should be cleared for on-the-fly Burst Chop mode. DDR_SDRAM_CFG[32-BE] should also be cleared for on-the-fly Burst Chop mode
AP_EN 5	0	**Address Parity Enable** Determines whether to generate and check address parity for the address and control signals when using registered DIMMs. If address parity is used, the MAPAR_OUT and MAPAR_IN pins are used to drive the parity bit and to receive errors from the open-drain parity error signal. Even parity are used, and parity is generated for the MA[15–0], MBA[2–0], $\overline{\text{MRAS}}$, $\overline{\text{MCAS}}$, $\overline{\text{MWE}}$ signals. Parity is not generated for the MCKE[0–1], MODT[0–1], or $\overline{\text{MCS}}$[0–1] signals. **Note:** Address parity should not be used for non-zero values of TIMING_CFG_3[CNTL_ADJ].	0 Address parity not used 1 Address parity used
D_INIT 4	0	**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 automatically initializes DRAM after it is enabled. This bit is automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle..	0 No data initialization, and no data initialization is scheduled. 1 The memory controller initializes memory when it is enabled.
— 3	0	Reserved. Write to zero for future compatibility.	
RCW_EN 2	0	**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 writes the register control words after DDR_SDRAM_CFG[MEM_EN] is set, unless DDR_SDRAM_CFG[BI] is set. The register control words are written with the values in DDR_SDRAM_RCW_1 and DDR_SDRAM_RCW_2.	0 Register control words are not automatically written during DRAM initialization. 1 Register control words are 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.
— 1	0	Reserved. Write to zero for future compatibility.	

Table 12-29. DDR_SDRAM_CFG_2 Field Descriptions (Continued)

Bit	Reset	Description	
MD_EN 0	0	**Mirrored DIMM Enable** Some DDR3 DIMMs are mirrored, where certain MA and MBA pins are mirrored on one side of the DIMM. When this bit is set, the controller knows to swap these signals before transmitting to the DRAM. The controller assumes that CS1 and CS3 are the 'mirrored' ranks of memory. The following signals are mirrored (MnBA0 versus MBA1; MA3 versus MA4; MA5 versus MA6; MA7 versus MA8).	0 Mirrored DIMMs are not used. 1 Mirrored DIMMs are used.

12.8.10 DDR SDRAM Mode Configuration Register (MnDDR_SDRAM_MODE)

DDR_SDRAM_MODE DDR SDRAM Mode Configuration Register Offset 0x0118

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

DDR_SDRAM_MODE sets the values loaded into the DDR mode registers.

Table 12-30. DDR_SDRAM_MODE Bit Descriptions

Bit	Refresh	Description
ESDMODE 31–16	0	**Extended SDRAM Mode** Specifies the initial value loaded into the DDR SDRAM extended 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, MA0 presents the LSB of ESDMODE, which corresponds to DDR_SDRAM_MODE bit 16. The MSB of the SDRAM extended mode register value must be stored at DDR_SDRAM_MODE bit 31 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.
SDMODE 15–0	0	**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, MA0 presents the LSB of SDMODE, which, corresponds to DDR_SDRAM_MODE bit 0. The MSB of the SDRAM mode register value must be stored at DDR_SDRAM_MODE bit 15. Because the memory controller forces SDMODE[8] to certain values depending upon the state of the initialization sequence (for resetting the SDRAM DLL) which is mapped to MA8; the memory controller ignores the corresponding bits of this field.

12.8.11 DDR SDRAM Mode Configuration 2 Register (MnDDR_SDRAM_MODE_2)

DDR_SDRAM_MODE_2 DDR SDRAM Mode Configuration 2 Register — Offset 0x011C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ESDMODE2															
Type	R/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**
	ESDMODE3															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_MODE_2 sets the values loaded into the DDR extended mode 2 and 3 registers.

Table 12-31. DDR_SDRAM_MODE_2 Bit Descriptions

Bit	Reset	Description
ESDMODE2 31–16	0	**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, MA0 presents the LSB bit of ESDMODE2, which corresponds to DDR_SDRAM_MODE_2 bit 16. The MSB of the SDRAM extended mode 2 register value must be stored at DDR_SDRAM_MODE_2 bit 31.
ESDMODE3 15–0	0	**Extended SDRAM Mode 3** Specifies the initial value loaded into the DDR SDRAM Extended 3 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, MA0 presents the LSB of ESDMODE3, which corresponds to DDR_SDRAM_MODE_2 bit 0. The MSB of the SDRAM extended mode 3 register value must be stored at DDR_SDRAM_MODE_2 bit 15.

12.8.12 DDR SDRAM Mode Control Register (MnDDR_SDRAM_MD_CNTL)

DDR_SDRAM_MD_CNTL DDR SDRAM Mode Control Register — Offset 0x0120

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MD_ EN	—	CS_ SEL		—	MD_SEL			SET_ REF	SET_ PRE	CKE_CNTL		WRC W	—		
Type	R/W	R	R/W		R	R/W							R			
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**
	MD_VALUE															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_MD_CNTL allows software to force mode/extended mode register set commands to DRAM, including the following:

- 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

Note: Each command initiated through the DDR_SDRAM_MD_CNTL register should be initiated separately in the order required by the DDR SDRAM. If multiple commands are initiated simultaneously, the execution order is not guarantied and can cause malfunction of the DDR SDRAM.

Table 12-32. DDR_SDRAM_MD_CNTL Bit Descriptions

Bit	Reset	Description	Settings
MD_EN 31	0	**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: • 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.
— 30	0	Reserved. Write to zero for future compatibility.	
CS_SEL 29–28	0	**Select Chip Select** Specifies the chip select to drive active due to any command forced by software in DDR_SDRAM_MD_CNTL.	00 Chip select 0 is active. 01 Chip select 1 is active. 10 Reserved. 11 Reserved.
— 27	0	Reserved. Write to zero for future compatibility.	
MD_SEL 26–24	000	**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 is presented onto the memory bank address pins (MBA*n*) of the DDR controller. 000 MR 001 EMR 010 EMR2 011 EMR3	000 MR Mode register 001 EMR Extended Mode Register 010 EMR2 Extended Mode Register 2 011 EMR3 Extended Mode Register 3
SET_REF 23	0	**Set Refresh** Forces an immediate refresh to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. Software sets this bit and hardware clears it when the command is issued. **Note:** SET_REF, and SET_PRE are mutually exclusive; they cannot be set at the same time	0 No REFRESH command to issue. 1 A REFRESH command is ready to issue.

Table 12-32. DDR_SDRAM_MD_CNTL Bit Descriptions (Continued)

Bit	Reset	Description	Settings
SET_PRE 22	0	**Set Precharge** Forces a precharge command or precharge all command to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. Software sets this bit, and hardware clears it when the command is issued. **Note:** SET_REF, and SET_PRE are mutually exclusive; they cannot be set at the same time	0 No PRECHARGE ALL command to issue. 1 Issue a PRECHARGE or PRECHARGE ALL command.
CKE_CNTL 21–20	0	**Clock Enable Control** Software uses tthese bits to set or clear all CKE signals issued to DRAM. When software forces the value driven on CKE, that value continues to be forced until software clears the CKE_CNTL bits. The DDR controller continues to drive the CKE signals to the value forced by software until another event causes the CKE signals to change (that is, self refresh entry/exit, power down entry/exit).	00 Software does not force the CKE signals. 01 Software forces the CKE signals to a low value. 10 Software forces the CKE signals to a high value. 11 Reserved.
WRCW 19	0	**Write Register Control Word** If software sets this bit, then a register control word is written by asserting the selected chip selects while providing the programmed data on the MA and MBA signals. $\overline{RAS}$, $\overline{CAS}$, and $\overline{WE}$ 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 DIMMs 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 is 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 is not issued if MD_EN is set. 1 Indicates that a register control word write is issued if MD_EN is set.
— 18–16	0	Reserved. Write to zero for future compatibility.	
MD_ VALUE 15–0	0	**Mode Register Value** This field specifies the value that is presented on the memory address pins of the DDR controller during a MODE REGISTER SET command. This field 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, MD_VALUE10 is mapped to MA10 to distinguish between a PRECHARGE command and a PRECHARGE ALL command, as follows:	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 All other values are not valid.

Table 12-33 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 12-33. 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	—
SET_PRE	0	0	1	—
CS_SEL	Chooses chip select (CS)			—
MD_SEL	Select mode register. See **Table 12-32.**	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only MD_VALUE10 is significant. See **Table 12-32.**	—
CKE_CNTL	0	0	0	See **Table 12-32.**

12.8.13 DDR SDRAM Interval Configuration Register (MnDDR_SDRAM_INTERVAL)

DDR_SDRAM_INTERVAL DDR SDRAM Interval Configuration Register — Offset 0x0124

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	REFINT															
Type	R/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**
	—		BSTOPRE													
Type			R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_INTERVAL sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, it specifies the number of DRAM cycles that a page is maintained after it is accessed.

Table 12-34. DDR_SDRAM_INTERVAL Bit Descriptions

Bit	Refresh	Description
REFINT 31–16	0	**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 are not issued when REFINT is cleared to all 0s.

Table 12-34. DDR_SDRAM_INTERVAL Bit Descriptions (Continued)

Bit	Refresh	Description
— 15–14	0	Reserved. Write to zero for future compatibility.
BSTOPRE 13–0	0	**Precharge Interval** Specifies the duration (in memory bus clock cycles) that a page is retained after a DDR SDRAM access. The page open counter is loaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with an SDRAM precharge bank command as soon as possible. If BSTOPRE has a value of 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.

12.8.14 DDR SDRAM Data Initialization Register (MnDDR_DATA_INIT)

DDR_DATA_INIT DDR SDRAM Data Initialization Register Offset 0x0128

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

DDR_DATA_INIT provides the value to initialize memory if DDR_SDRAM_CFG_2[D_INIT] is set. When ECC is enabled, initializing all the available memory space can ensure that reads from unwritten addresses does not return ECC errors and interrupts.

Table 12-35. DDR_DATA_INIT Bit Descriptions

Bits	Reset	Description
INIT_VALUE 31–0	0	**Initialization Value** Specifies the initialization value for the DRAM if DDR_SDRAM_CFG_2[D_INIT] is set.

12.8.15 DDR SDRAM Clock Control Configuration Register (MnDDR_SDRAM_CLK_CNTL)

DDR_SDRAM_CLK_CNTL DDR SDRAM Clock Control Configuration Register Offset 0x0130

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—					CLK_ADJUST				—						
Type	R					R/W				R						
Reset	0	0	0	0	0	0	1	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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_CLK_CNTL provides a source synchronous option, along with a 1/8 cycle clock adjustment.

Table 12-36. DDR_SDRAM_CLK_CNTL Bit Descriptions

Bit	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	
CLK_ADJUST 26–23	4'b0100	**Clock Adjust** Specifies when the clock is launched in relationship to the address/command. Set delay from address/command start timing to MCK rising edge. **Figure 12-5**, **Figure 12-6**, and **Figure 12-7** in **Section 12.4** show the timing relationships between clock and command for the case of CLK_ADJUST 0100 1/2 clock delay, which means nominal timing. The MCK rising edge is in the center of the address/command period in general.	0000 Clock launched and aligned with address/command. 0001 Clock launched 1/8 applied cycle after address/command. 0010 Clock launched 1/4 applied cycle after address/command. 0011 Clock launched 3/8 applied cycle after address/command. 0100 Clock launched 1/2 applied cycle after address/command. 0101 Clock launched 5/8 applied cycle after address/command. 0110 Clock launched 3/4 applied cycle after address/command. 0111 Clock launched 7/8 applied cycle after address/command. 1000 Clock launched 1 applied cycle after address/command. 1001–1111Reserved.
— 22–0	0	Reserved. Write to zero for future compatibility.	

12.8.16 DDR SDRAM Initialization Address Register (MnDDR_INIT_ADDR)

DDR_INIT_ADDR DDR SDRAM Initialization Address Register Offset 0x0148

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

DDR_INIT_ADDR provides the address for the data strobe to data skew adjustment and automatic CAS to preamble calibration after setting DDR_SDRAM_CFG[MEM_EN].

Note: After the skew adjustment, this address contains bad ECC data, which is not important at power-on reset because all memory should subsequently be initialized if ECC is enabled either through software or through the use of the DDR_SDRAM_CFG_2[D_INIT] bit. However, if the DSP is reset after the DRAM enters Self-Refresh mode, memory is not initialized. Therefore this address should be written to avoid possible ECC errors when this address is accessed later.

Table 12-37. DDR_INIT_ADDR Bit Descriptions

Bit	Reset	Description	Settings
INIT_ADDR 31–0	0	**Initialization Address** Provides the address used for the data to data strobes skew adjustment and automatic CAS to preamble calibration after setting DDR_SDRAM_CFG[MEM_EN]. This address is written during the initialization sequence.	

12.8.17 DDR Initialization Enable (MnDDR_INIT_EN)

DDR_INIT_EN — DDR SDRAM Initialization Enable — Offset 0x014C

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

The DDR SDRAM initialization enable register provides the enable bit to use the address given by DDR_INIT_ADDR[INIT_ADDR] for the data to data strobes deskew calibration and for automatic CAS to preamble calibration during the DDR SDRAM initialization. **Table 12-38** describes the DDR_INIT_EN fields.

Table 12-38. DDR_INIT_EN Field Descriptions

Bit	Reset	Description	Settings
UIA 31	0	**Use Initialization Address** Indicates whether to use the initialization address.	0 Use the default address for training sequence as calculated by the controller. This is the first valid address in the first enabled chip select. 1 Use the initialization address programmed in DDR_INIT_ADDR.
— 30–0	0	Reserved. Write to zero for future compatibility.	

12.8.18 DDR SDRAM Timing Configuration 4 (MnTIMING_CFG_4)

TIMING_CFG_4 DDR SDRAM Timing Con figuration 4 Offset 0x0160

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RWT				WRT				RRT				WWT			
Type	R/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**
	—															DLL_LOCK
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DDR SDRAM timing configuration 4 register provides additional timing fields required to support DDR3 memories. **Table 12-39** describes the TIMING_CFG_4 fields.

Table 12-39. TIMING_CFG_4 Field Descriptions

Bits	Reset	Description	Settings
RWT 31–28	0	**Read-to-Write Turnaround for Same Chip Select** Specifies how many cycles are 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 uses 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] is also 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 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
WRT 27–24	0	**Write-to-Read Turnaround for Same Chip Select** Specifies how many cycles are 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 uses 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] is also 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

Table 12-39. TIMING_CFG_4 Field Descriptions (Continued)

Bits	Reset	Description	Settings
RRT 23–20	0	**Read-to-Read Turnaround for Same Chip Select** Specifies how many cycles are added between reads to the same chip select. If a value of 0000 is chosen, then 2 cycles is required between read commands to the same chip select if 4-beat bursts are used (4 cycles are 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 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
WWT 19–16	0	**Write-to-Write Turnaround for Same Chip Select** Specifies how many cycles are added between writes to the same chip select. If a value of 0000 is chosen, then 2 cycles is required between write commands to the same chip select if 4-beat bursts are used (4 cycles are 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
— 15–2	0	Reserved. Write to zero for future compatibility.	
DLL_LOCK 1–0	0	**DDR SDRAM DLL Lock Time** This provides the number of cycles that it takes for the DRAMs DLL to lock at power-on reset and after exiting self refresh. The controller waits 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

12.8.19 DDR SDRAM Timing Configuration 5 (MnTIMING_CFG_5)

TIMING_CFG_5 DDR SDRAM Timing Con figuration 5 Offset 0x0164

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			RODT_ON					—	RODT_OFF			—			WODT_ON
Type	R/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**
	WODT_ON				—	WODT_OFF			—							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DDR SDRAM timing configuration 5 register provides additional timing fields required to support DDR3 memories. **Table 12-40** describes the TIMING_CFG_5 fields.

Table 12-40. TIMING_CFG_5 Field Descriptions

Bits	Reset	Description	Settings
— 31–29	0	Reserved. Write to zero for future compatibility.	
RODT_ON 28–24	0	**Read-to-ODT On** Specifies the number of cycles that passes 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 RL - 3 cycles to support legacy of past products. RL is the read latency, derived from CAS latency + additive latency. If 2T/3T timing is used, one/two extra cycles are automatically added to the value selected in this field.	00000 RL - 3 clocks 00001 0 clocks 00010 1 clocks 00011 2 clocks 00100 3 clocks 00101 4 clocks 00110 5 clocks 00111 6 clocks 01000 7 clocks 01001 8 clocks 01010 9 clocks 01011 10 clocks 01100 11 clocks 01101 12 clocks 01110 13 clocks. 011111 14 clocks 10000 15 clocks 10001 16 clocks 10010 17 clocks 10011 18 clocks 10100 19 clocks 10101 20 clocks 10110 21 clocks 10111 22 clocks 11000 23 clocks 11001 24 clocks 11010 25 clocks 11011 26 clocks 11100 27 clocks 11101 28 clocks 11110 29 clocks. 11111 30 clocks

Table 12-40. TIMING_CFG_5 Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 23	0	Reserved. Write to zero for future compatibility.	
RODT_OFF 22–20	0	**Read to ODT Off** Specifies the number of cycles that the relevant ODT signal(s) remains asserted for each read transaction. The default case (000) leaves the ODT signal(s) asserted for 3 DRAM cycles.	000 3 clocks 001 1 clock 010 2 clocks 010 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
— 19–17	0	Reserved. Write to zero for future compatibility.	
WODT_ON 16–12	0	**Write-to-ODT On** Specifies the number of cycles that passes 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 WL - 3 cycles to support legacy of past products. WL is the write latency, derived from Write Latency + Additive Latency. If 2T/3T timing is used, one/two extra cycles are automatically added to the value selected in this field.	00000 WL - 3 clocks 00001 0 clocks 00010 1 clocks 00011 2 clocks 00100 3 clocks 00101 4 clocks 00110 5 clocks 00111 6 clocks 01000 7 clocks 01001 8 clocks 01010 9 clocks 01011 10 clocks 01100 11 clocks 01101 12 clocks 01110 13 clocks. 011111 14 clocks 10000 15 clocks 10001 16 clocks 10010 17 clocks 10011 18 clocks 10100 19 clocks 10101 20 clocks 10110 21 clocks 10111 22 clocks 11000 23 clocks 11001 24 clocks 11010 25 clocks 11011 26 clocks 11100 27 clocks 11101 28 clocks 11110 29 clocks. 11111 30 clocks
— 11	0	Reserved. Write to zero for future compatibility.	

Table 12-40. TIMING_CFG_5 Field Descriptions (Continued)

Bits	Reset	Description	Settings
WODT_OFF 10–8	0	**Write to ODT Off** Specifies the number of cycles that the relevant ODT signal(s) remains asserted for each write transaction. The default case (000) leaves the ODT signal(s) asserted for 3 DRAM cycles.	000 3 clocks 001 1 clock 010 2 clocks 010 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
— 7–0	0	Reserved. Write to zero for future compatibility.	

12.8.20 DDR ZQ Calibration Control (MnDDR_ZQ_CNTL)

DDR_ZQ_ CNTL DDR ZQ Calibration Control Offset 0x0170

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ZQ_EN	—			ZQINIT				—				ZQOPER			
Type	R/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**
	—				ZQCS				—							
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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])
```

it is possible that one refresh is skipped when the controller exits self refresh. If this is an issue, then posted refreshes can 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 do not allow missed refresh cycles. **Table 12-41** describes the DDR_ZQ_CNTL fields.

Table 12-41. DDR_ZQ_CNTL Field Descriptions

Bit	Reset	Description	Settings
ZQ_EN 31	0	**ZQ Calibration Enable** This bit determines whether ZQ calibration is used. This bit should be set only if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 0b111).	0 ZQ Calibration is not used. 1 ZQ Calibration is used. A ZQCL command is issued by the DDR controller after power-on reset and anytime the DDR controller exits self refresh. A ZQCS command is issued every 32 refresh sequences to account for VT variations.
— 30–28	0	Reserved. Write to zero for future compatibility.	
ZQ_INIT 27–24	0	**Power-on Reset ZQ Calibration Time (t_{ZQinit})** Determines the number of cycles that must be allowed for DRAM ZQ calibration at power-on reset. Each chip select is calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command can be issued.	0111 128 clocks 1000 256 clocks 1001 512 clocks 1010 1024 clocks All other values reserved.
— 23–20	0	Reserved. Write to zero for future compatibility.	
ZQOPER 19–16	0	**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 is calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued.	0111 128 clocks 1000 256 clocks 1001 512 clocks 1010 1024 clocks All other values reserved.
— 15–12	0	Reserved. Write to zero for future compatibility.	
ZQCS 11–8	0	**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 is 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 All other values reserved.
— 7–0	0	Reserved. Write to zero for future compatibility.	

12.8.21 DDR Write Leveling Control (MnDDR_WRLVL_CNTL)

DDR_WRLVL_CNTL DDR Write Leveling Control Offset 0x0174

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRLVL_EN	—				WRLVL_MRD			—	WRLVL_ODTEN			—	WRLVL_DQSEN		
Type	R/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**
	WRLVL_SMPL				—	WRLVL_WLR			—				WRLVL_START			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DDR Write Leveling Control register provides controls for write leveling, as it is supported for DDR3 memory devices. **Table 12-42** describes the DDR_WRLVL_CNTL fields.

Table 12-42. DDR_WRLVL_CNTL Field Descriptions

Bits	Reset	Description	Settings
WRLVL_EN 31	0	**Write Leveling Enable** This bit determines if write leveling is used. If this bit is set, then the DDR controller performs write leveling immediately after initializing the DRAM. This bit should only be set if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 3'b111). In addition, write leveling is not supported for DDR3 mirrored DIMMs.	0 Write leveling is not used 1 Write leveling is used
— 30–27	0	Reserved. Write to zero for future compatibility.	
WRLVL_MRD 26–24	0	**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 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
— 23	0	Reserved. Write to zero for future compatibility.	
WRLVL_ODTEN 22–20	0	**ODT Delay after Margining Mode Is Programmed (t_{WL_ODTEN})** Determines how many cycles to wait after margining mode is programmed.until ODT is 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
— 19	0	Reserved. Write to zero for future compatibility.	

Table 12-42. DDR_WRLVL_CNTL Field Descriptions (Continued)

Bits	Reset	Description	Settings
WRLVL_ DQSEN 18–16	0	**DQS/$\overline{\text{DQS}}$ 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
WRLVL_ SMPL 15–12	0	**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 Reserved (if DDR_ WRLVL_CNTL[WRL VL_EN] is set) 0001 1 clocks 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 1010 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1010 14 clocks 1111 15 clocks
— 11	0	Reserved. Write to zero for future compatibility.	
WRLVL_ WLR 10–8	0	**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

Table 12-42. DDR_WRLVL_CNTL Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 7–5	0	Reserved. Write to zero for future compatibility.	
WRLVL_START 4–0	0	**Write Leveling Start Time** Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled.	00000 0 clock delay 00001 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101– 11111 Reserved

12.8.22 DDR Write Leveling Control 2 (MnDDR_WRLVL_CNTL_2)

DDR_WRLVL_CNTL_2 DDR Write Leveling Control 2 Offset 0x0190

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			WRLVL_START_1					—			WRLVL_START_2				
Type	R/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**
	—			WRLVL_START_3					—			WRLVL_START_4				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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. **Table 12-43** describes the DDR_WRLVL_CNTL_2 fields.

Table 12-43. DDR_WRLVL_CNTL_2 Field Descriptions

Bits	Reset	Description	Settings
— 31–29	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_1 28–24	0	**Write Leveling Start Time for DQS1** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101–11111 Reserved

Table 12-43. DDR_WRLVL_CNTL_2 Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 23–21	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_2 20–16	0	**Write Leveling Start Time for DQS2** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101– 11111 Reserved
— 15–13	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_3 12–8	0	**Write Leveling Start Time for DQS3** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101– 11111 Reserved

Table 12-43. DDR_WRLVL_CNTL_2 Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 7–5	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_4 4–0	0	**Write Leveling Start Time for DQS4** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101– 11111 Reserved

12.8.23 DDR Write Leveling Control 3 (MnDDR_WRLVL_CNTL_3)

DDR_WRLVL_CNTL_3 DDR Write Leveling Control 3 Offset 0x0194

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			WRLVL_START_5					—			WRLVL_START_6				
Type	R/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**
	—			WRLVL_START_7					—			WRLVL_START_8				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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. **Table 12-43** describes the DDR_WRLVL_CNTL_3 fields.

Table 12-44. DDR_WRLVL_CNTL_3 Field Descriptions

Bits	Reset	Description	Settings
— 31–29	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_5 28–24	0	**Write Leveling Start Time for DQS5** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101– 11111 Reserved
— 23–21	0	Reserved. Write to zero for future compatibility.	

Table 12-44. DDR_WRLVL_CNTL_3 Field Descriptions (Continued)

Bits	Reset	Description	Settings
WRLVL_START_6 20–16	0	**Write Leveling Start Time for DQS6** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101–11111 Reserved
— 15–13	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_7 12–8	0	**Write Leveling Start Time for DQS7** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101–11111 Reserved

Table 12-44. DDR_WRLVL_CNTL_3 Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 7–5	0	Reserved. Write to zero for future compatibility.	
WRLVL_START_8 4–0	0	**Write Leveling Start Time for DQS8** 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 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101– 11111 Reserved

12.8.24 DDR Pre-Drive Conditioning Control (MnDDR_PD_CNTL)

DDR_PD_CNTL DDR Pre_Drive Conditioning Control Offset 0x0178

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PD_EN	TVPD			—	PDAR							—	PDAW		
Type	R/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**
	PDAW				—	PD_ON					—	PD_OFF				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DDR Pre-Drive Conditioning Control register provides controls for asserting termination at the on-chip I/Os prior to driving DQS/DQ for write commands. **Table 12-45** describes the DDR_PD_CNTL fields.

Table 12-45. DDR_PD_CNTL Field Descriptions

Bits	Reset	Description	Settings
PD_EN 31	0	**Pre-Drive Conditioning Enable** This bit determines whether termination is asserted to the on-chip I/Os prior to write commands. This helps bring DRAM/IO terminated signals near V_{REF} before driving an active preamble. The result is a more symmetrical eye with less overshoot on the first rising edge of these signals.	0 Pre-drive conditioning is not used 1 Pre-drive conditioning is used
TVPD 30–28	0	**Termination Value During Pre-drive Conditioning** This represents the value of the termination that is used during pre-drive conditioning.	000 75 ohms 001 60 ohms 010 150 ohms 011 120 ohms 100 46 ohms 101 43 ohms 110 33 ohms 111 Reserved
— 27	0	Reserved. Write to zero for future compatibility.	
PDAR 26–20	0	**Pre-Drive After Read** If pre-drive conditioning is enabled, it may not be desirable to assert termination to the on-chip IOs immediately after a prior read. This field represents the number of cycles that must pass after a previous read is issue before pre-drive conditioning is available for a future write. Note that the decision to use pre-drive conditioning is made at the time the write command is issued.	0000000 0 clocks 0000001 1 clocks 0000010 2 clocks ... 1111111 127 clocks **Note:** The value represents the actual number of clocks to use
— 19	0	Reserved. Write to zero for future compatibility.	

Table 12-45. DDR_PD_CNTL Field Descriptions (Continued)

Bits	Reset	Description	Settings
PDAW 18–12	0	**Pre-Drive After Write** If pre-drive conditioning is enabled, it may not be desirable to assert termination to the on-chip IOs immediately after a prior write. This field represents the number of cycles that must pass after a previous write is issue before pre-drive conditioning is available for a future write. Note that the decision to use pre-drive conditioning is made at the time the write command is issued.	0000000 0 clocks 0000001 1 clocks 0000010 2 clocks ... 1111111 127 clocks **Note:** The value represents the actual number of clocks to use
— 11	0	Reserved. Write to zero for future compatibility.	
PD_ON 10–6	0	**Pre-Drive Conditioning On** Specifies the number of cycles that passes from when a write command is placed on the DRAM bus until the assertion of termination at the on-chip IOs. Note that DDR_PD_CNTL[PD_ON] + DDR_PD_CNTL[PD_OFF} should not be greater than 31.	00000 Reserved 00001 1 clocks 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 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

Table 12-45. DDR_PD_CNTL Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 5	0	Reserved. Write to zero for future compatibility.	
PD_OFF 4–0	0	**Pre-Drive Conditioning Off** Specifies the number of cycles that the termination at the on-chip IOs remains asserted prior to a write transaction. Note that DDR_PD_CNTL[PD_ON] + DDR_PD_CNTL[PD_OFF} should note be greater than 31.	00000 Reserved 00001 1 clocks 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 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

12.8.25 DDR Self Refresh Counter (MnDDR_SR_CNTR)

DDR_SR_CNTR DDR Self Refresh Counter Offset 0x017C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												SR_IT			
Type	R												R/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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SR_CNTR can be programmed to force the DDR controller to enter self refresh after a predefined period of idle time.

Table 12-46. DDR_SR_CNTR Bit Descriptions

Bit	Reset	Description	Settings	
— 31–20	0	Reserved. Write to zero for future compatibility.		
SR_IT 19–16	0	**Self Refresh Idle Threshold** Defines the number of DRAM cycles that must pass while the DDR controller is idle before it enters self refresh. Anytime a transaction is issued to the DDR controller, it resets its internal counter. When a new transaction is received by the DDR controller, it exits self refresh and reset its internal counter. If this field is zero, then the described power savings feature is disabled. In addition, if a non-zero value is programmed into this field, then the DDR controller exits self refresh anytime a transaction is issued to the DDR controller, regardless of the reason self refresh was initially entered.	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100-1111	Automatic self refresh entry disabled 2^{10} DRAM clocks 2^{12} DRAM clocks 2^{14} DRAM clocks 2^{16} DRAM clocks 2^{18} DRAM clocks 2^{20} DRAM clocks 2^{22} DRAM clocks 2^{24} DRAM clocks 2^{26} DRAM clocks 2^{28} DRAM clocks 2^{30} DRAM clocks Reserved
— 15–0	0	Reserved. Write to zero for future compatibility.		

12.8.26 DDR SDRAM Register Control Words 1 (MnDDR_SDRAM_RCW_1)

DDR_SDRAM_RCW_1 DDR SDRAM Register Control Word 1 Offset 0x0180

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RCW0				RCW1				RCW2				RCW3			
Type	R/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**
	RCW4				RCW5				RCW6				RCW7			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_RCW_1 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 is placed on MA3, MA4, MBA0, and MBA1 during register control word writes.

Table 12-47. DDR_SDRAM_RCW_1 Bit Descriptions

Bit	Reset	Description
RCW0 31–28	0	**Register Control Word 0** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 0.
RCW1 27–24	0	**Register Control Word 1** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 1.
RCW2 23–20	0	**Register Control Word 2** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 2.
RCW3 19–16	0	**Register Control Word 3** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 3.
RCW4 15–12	0	**Register Control Word 4** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 4.
RCW5 11–8	0	**Register Control Word 5** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 5.
RCW6 7–4	0	**Register Control Word 6** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 6.
RCW7 3–0	0	**Register Control Word 7** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 7.

12.8.27 DDR SDRAM Register Control Words 2 (MnDDR_SDRAM_RCW_2)

DDR_SDRAM_RCW_2 DDR SDRAM Register Control Word 2 Offset 0x0184

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RCW8				RCW9				RCW10				RCW11			
Type	R/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**
	RCW12				RCW13				RCW14				RCW15			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_SDRAM_RCW_2 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 is placed on MA3, MA4, MBA0, and MBA1 during register control word writes.

Table 12-48. DDR_SDRAM_RCW_2 Bit Descriptions

Bit	Reset	Description
RCW8 31–28	0	**Register Control Word 8** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 8.
RCW9 27–24	0	**Register Control Word 9** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 9.
RCW10 23–20	0	**Register Control Word 10** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 10.
RCW11 19–16	0	**Register Control Word 11** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 11.
RCW12 15–12	0	**Register Control Word 12** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 12.
RCW13 11–8	0	**Register Control Word 13** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 13.
RCW14 7–4	0	**Register Control Word 14** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 14.
RCW15 3–0	0	**Register Control Word 15** Represents the value that is placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 15.

12.8.28 DDR Debug Status Register 1 (MnDDRDSR_1)

DDRDSR_1 DDR Debug Status Register 1 Offset 0x0B20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DDRDC		MDICPZ				MDICNZ				—					
Type	R															
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**
	CPZ				CNZ				DPZ				DNZ			
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRDSR_1 contains the DDR driver compensation input value and the current settings of the P and N FET impedance for MDICn, command/control, and data. Note: This register is read only

Table 12-49. DDRDSR_1 Bit Descriptions

Bit	Reset	Description
DDRDC 31–30	0	**DDR Driver Compensation Input Value**
MDICPZ 29–26	0	**Current Setting of PFET Driver MDIC Impedance**
MDICNZ 25–22	0	**Current Setting of NFET Driver MDIC Impedance**
— 21–16	0	Reserved. Always read as zero
CPZ 15–12	0	**Current Setting of PFET Driver Command Impedance**
CNZ 11–8	0	**Current Setting of NFET Driver Command Impedance**
DPZ 7–4	0	**Current Setting of PFET Driver Data Impedance**
DNZ 3–0	0	**Current Setting of NFET Driver Data Impedance**

12.8.29 DDR Debug Status Register 2 (MnDDRDSR_2)

DDRDSR_2 DDR Debug Status Register 2 Offset 0x0B24

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLKPZ				CLKNZ				—							
Type	R															
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**
	—															
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRDSR_2 contains the current settings of the P and N FET impedance for the DDR drivers for clocks. Note: This register is read only

Table 12-50. DDRDSR_2 Bit Descriptions

Bit	Reset	Description
CLKPZ 31–28	0	**Current Setting of PFET Driver Clock Impedance**
CLKNZ 27–24	0	**Current Setting of NFET Driver Clock Impedance**
— 23–0	0	Reserved. Always read as zero

12.8.30 DDR Control Driver Register 1 (MnDDRCDR_1)

DDRCDR_1 DDR Control Driver Register 1 Offset 0x0B28

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DHC_EN	DSO_MDIC_EN	DSO_MDICPZ				DSO_MDICNZ				DSO_MDIC_PZ_OE	DSO_MDIC_NZ_OE	ODT		DSO_C_EN	DSO_D_EN
Type	R/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**
	DSO_CPZ				DSO_CNZ				DSO_DPZ				DSO_DNZ			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRCDR_1 sets the driver hardware compensation enable, the DDR MDIC driver P/N impedance, ODT termination value for IOs, driver software override enable for MDIC, driver software override enable for address/command, driver software override enable for data, the DDR address/command driver P/N impedance, and the DDR data driver P/N impedance.

Table 12-51. DDRCDR_1 Bit Descriptions

Bit	Reset	Description	Settings
DHC_EN 31	0	**DDR Driver Hardware Compensation Enable**	0 Hardware compensation disabled. 1 Hardware compensation enabled.
DSO_MDIC_EN 30	0	**Driver Software Override Enable for MDIC**	0 Software override for MDIC disabled. 1 Software override for MDIC enabled.
DSO_MDICPZ 29–26	0	**Driver Software Override Value for MDIC P-Impedance**	
DSO_MDICNZ 25–22	0	**Driver Software Override Value for MDIC N-Impedance**	
DSO_MDIC_PZ_OE 21	0	**Driver Software Override Output Enable for P-Impedance -** This field is effective only if DSO_MDIC_EN is set	0 Output disabled. 1 Output enabled.
DSO_MDIC_NZ_OE 20	0	**Driver Software Override Output Enable for N-Impedance -** This field is effective only if DSO_MDIC_EN is set	0 Output disabled. 1 Output enabled.
ODT 19–18	0	**ODT Termination Value for I/Os** This is combined with DDRCDR_2[ODT] to determine the termination value. The termination value is based on concatenating these 2 fields (DDRCDR_1[ODT]|DDRCDR_2[ODT])	000 75 ohm (JEDEC DDR2) 001 55 ohm 010 60 ohm (JEDEC DDR3) 011 50 ohm (JEDEC DDR2) 100 150 ohm (JEDEC DDR2) 101 43 ohm 110 120 ohm (JEDEC DDR3) 111 Reserved
DSO_C_EN 17	0	**Driver Software Override Enable for Address/Command**	0 Override disabled. 1 Override enabled.
DSO_D_EN 16	0	**Driver Software Override Enable for Data**	0 Override disabled. 1 Override enabled.
DSO_CPZ 15–12	0	**DDR Driver Software Override Value for Command P-Impedance Override.**	
DSO_CNZ 11–8	0	**DDR Driver Software Override Value for Command N-Impedance Override.**	
DSO_DPZ 7–4	0	**DDR Driver Software Override Value for Data P-Impedance Override.**	
DSO_DNZ 3–0	0	**DDR Driver Software Override Value for Data N-Impedance Override.**	

The fields in DDRCDR_1, other than DDRCDR_1[ODT], are used to enable driver calibration with the MDIC[0–1] pins. This can be used to calibrate the DDR drivers to 18 ohms. However, this should only be used for full-strength driver applications. If half strength is desired, then this calibration should remain disabled. The hardware DDR driver calibration is enabled via DDRCDR_1[DHC_EN].

Note: All Driver Calibration setting, either software or hardware, should be set before DDR_SDRAM_CFG[MEM_EN] is set.

Software can be used to calibrate the drivers instead of the automatic hardware calibration. If software calibration is used, the following steps should be taken:

1. Set DDRCDR_1[DSO_MDIC_EN] and ensure that DDRCDR_1[DHC_EN] is cleared
2. Set the highest impedance (value 0000) for DDRCDR_1[DSO_MDICPZ]
3. Set DDRCDR_1[DSO_MDIC_PZ_OE] to enable the output enable for MDIC[0]
4. After at least 4 cycles, read DDRDSR_1[0]. If the value is 0, then use the next lowest impedance, and read DDRDSR_1[0] again. Once a value of 1 is detected, then leave DDRCDR_1[DSO_MDICPZ] at the calibrated value
5. Clear DDRCDR_1[DSO_MDIC_PZ_OE]
6. After DDRCDR_1[DSO_MDICPZ} is calibrated, set a value of 0000 for DDRCDR_1[DSO_MDICNZ]
7. Set DDRCDR_1[DSO_MDIC_NZ_OE] to enable the output enable for MDIC[1]
8. After at least 4 cycles, read DDRDSR_1[1]. If the value is 1, then use the next lowest impedance, and read DDRDSR_1[1] again. Once a value of 0 is detected, then leave DDRCDR_1[DSO_MDICNZ] at the calibrated value
9. Clear DDRCDR_1[DSO_MDIC_NZ_OE]

The legal impedance values (from highest impedance to lowest impedance) for DDR2 (1.8 V) are:

- 0000 lowest strength/highest impedance
- 0001
- 0011
- 0010
- 0110
- 0111 half strength - used when driver calibration is not used by setting DDR_SDRAM_CFG[HSE]
- 0101
- 0100
- 1100
- 1101
- 1111
- 1110
- 1010
- 1011 full strength (default)
- 1001
- 1000 highest strength/lowest impedance

The legal impedance values (from highest impedance to lowest impedance) for DDR3 (1.5 V) are:

- 0000 lowest strength/highest impedance
- 0001
- 0011
- 0010
- 0110
- 0111 half strength - used when driver calibration is not used by setting DDR_SDRAM_CFG[HSE]
- 0101
- 0100
- 1100
- 1101
- 1111
- 1110
- 1010
- 1011
- 1001
- 1000 highest strength/lowest impedance (default)

Note: Drivers may either be calibrated to full-strength or half-strength

12.8.31 DDR Control Driver Register 2 (MnDDRCDR_2)

DDRCDR_2 DDR Control Driver Register 2 Offset 0x0B2C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DSO_CLK_EN	—			DSO_CLKPZ				DSO_CLKNZ				—			
Type	R/W	R			R/W								R			
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**
	—															ODT
Type	R															R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRCDR_2 sets the driver software override enable for clocks, and the DDR clocks driver P/N impedance.

Table 12-52. DDRCDR_2 Bit Descriptions

Bit	Reset	Description	Settings
DSO_CLK_EN 31	0	**Driver Software Override Enable for Clocks**	0 Software override disabled. 1 Software override enabled.
— 30–28	0	Reserved. Write to zero for future compatibility.	
DSO_CLKPZ 27–24	0	**DDR Driver Software Override Value for Clocks P-Impedance Override**	
DSO_CLKNZ 23–20	0	**DDR Driver Software Override Value for Clocks N-Impedance Override**	
— 19–1	0	Reserved. Write to zero for future compatibility.	
ODT 0	0	**ODT Termination Value for I/Os** This is combined with DDRCDR_1[ODT] to determine the termination value. The termination value is based on concatenating these 2 fields (DDRCDR_1[ODT]|DDRCDR_2[ODT]).	000 75 ohm (JEDEC DDR2) 001 55 ohm 010 60 ohm (JEDEC DDR3) 011 50 ohm (JEDEC DDR2) 100 150 ohm (JEDEC DDR2) 101 43 ohm 110 120 ohm (JEDEC DDR3) 111 Reserved

12.8.32 DDR SDRAM IP Block Revision 1 Register (MnDDR_IP_REV1)

DDR_IP_REV1 DDR SDRAM IP Block Revision 1 Register Offset 0x0BF8

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

DDR_IP_REV1 provides read-only fields with the IP block ID, along with major and minor revision information.

Table 12-53. DDR_IP_REV1 Bit Descriptions

Bit	Reset	Description
IP_ID 31–16	0x0002	**IP Block ID** For the DDR controller.
IP_MJ 15–8	0x04	**Major Revision**
IP_MN 7–0	0x00	**Minor Revision**

12.8.33 DDR SDRAM IP Block Revision 2 Register (MnDDR_IP_REV2)

DDR_IP_REV2 DDR SDRAM IP Block Revision 2 Register Offset 0x0BFC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								IP_INIT							
Type	R															
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**
	—								IP_CFG							
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_IP_REV2 provides read-only fields with the IP block integration and configuration options.

Table 12-54. DDR_IP_REV2 Bit Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Always read as zero
IP_INT 23–16	0	**IP Block Integration Options**

Table 12-54. DDR_IP_REV2 Bit Descriptions (Continued)

Bit	Reset	Description
— 15–8	0	Reserved. Always read as zero
IP_CFG 7–0	0	**IP Block Configuration Options**

12.8.34 DDR SDRAM Memory Data Path Error Injection Mask High Register (MnDATA_ERR_INJECT_HI)

DATA_ERR_INJECT_HI DDR SDRAM Memory Data Path Error Injection Mask High Register — Offset 0x0E00

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

DATA_ERR_INJECT_HI register is used to inject ECC errors for the 32-msb of a 64-bit SDRAM data bus interfaces.

Table 12-55. DATA_ERR_INJECT_HI Bit Descriptions

Bit	Reset	Description
EIMH 31–0	0	**Error Injection Mask High Data Path** Tests ECC by forcing errors on the highest 32 bits of the data path. When error injection is enabled, setting a bit causes the corresponding data path bit to be inverted during memory bus writes.

12.8.35 DDR SDRAM Memory Data Path Error Injection Mask Low Register (MnDATA_ERR_INJECT_LO)

DATA_ERR_INJECT_LO DDR SDRAM Memory Data Path Error Injection Mask Low Register — Offset 0x0E04

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

DATA_ERR_INJECT_LO register is used to inject ECC errors for the 32-lsb of a 64-bit SDRAM data bus interfaces.

Table 12-56. DATA_ERR_INJECT_LO Bit Descriptions

Bit	Reset	Description
EIML 31–0	0	**Error Injection Mask Low Data Path** Tests ECC by forcing errors on the lowest 32 bits of the data path. When the Error Injection is enabled, setting a bit causes the corresponding data path bit to be inverted during memory bus writes.

12.8.36 DDR SDRAM Memory Data Path Error Injection Mask ECC Register (MnERR_INJECT)

ERR_INJECT DDR SDRAM Memory Data Path Error Injection Mask ECC Register — Offset 0x0E08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															APIEN
Type	R															R/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**
	—							EIEN	EEIM							
Type	R							R/W								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_INJECT register sets the ECC mask, enables errors to be written to ECC memory. In addition, a single address parity error can be injected through this register.

Table 12-57. ERR_INJECT Bit Descriptions

Bit	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
APIEN 15	0	**Address Parity Error Injection Enable**. This bit is cleared by hardware after a single address parity error has been injected.	0 Address parity error injection disabled. 1 Address parity error injection enabled.
— 15–9	0	Reserved. Write to zero for future compatibility.	
EIEN 8	0	**Error Injection Enable** Enables/disables injection of errors. This applies to the data mask bits and to the ECC mask bits. **Note:** Error injection should not be enabled until the memory controller has been enabled through DDR_SDRAM_CFG[MEM_EN].	0 Error injection disabled. 1 Error injection enabled.
EEIM 7–0	0	**ECC Error Injection Mask (0–7)** Setting a mask bit causes the corresponding ECC bit to be inverted during memory bus writes.	

12.8.37 DDR SDRAM Memory Data Path Read Capture Data High Register (MnCAPTURE_DATA_HI)

CAPTURE_DATA_HI DDR SDRAM Memory Data Path Read Capture Data High Register — Offset 0x0E20

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

CAPTURE_DATA_HI stores the high 32 bits of the read data path during error capture.

Table 12-58. CAPTURE_DATA_HI Bit Descriptions

Bit	Reset	Description
ECHD 31–0	0	**Error Capture High Data Path** Captures the high 32 bits of the data path when errors are detected.

12.8.38 DDR SDRAM Memory Data Path Read Capture Data Low Register (MnCAPTURE_DATA_LO)

CAPTURE_DATA_LO DDR SDRAM Memory Data Path Read Capture Data Low Register — Offset 0x0E24

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

CAPTURE_DATA_LO stores the low 32 bits of the read data path during error capture.

Table 12-59. CAPTURE_DATA_LO Bit Descriptions

Bit	Reset	Description
ECLD 31–0	0	**Error Capture Low Data Path** Captures the low 32 bits of the data path when errors are detected.

12.8.39 DDR SDRAM Memory Data Path Read Capture ECC Register (MnCAPTURE_ECC)

CAPTURE_ECC DDR SDRAM Memory Data Path Read Capture ECC Register Offset 0x0E28

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								ECE-32							
Type	R								R/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**
	—								ECE-64							
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_ECC stores the ECC syndrome bits on the data bus when an error is detected.

Table 12-60. CAPTURE_ECC Bit Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
ECE-64 23–16	0	**Error Capture ECC** In 32-bit bus mode, captures the ECC bits of the 1st 32-bits when errors are detected
— 15–8	0	Reserved. Write to zero for future compatibility.
ECE-64 7–0	0	**Error Capture ECC** In 64-bit bus mode, captures the ECC bits of entire bus when errors are detected. (0–7) In 32-bit bus mode. captures the ECC bits of the 2nd 32-bits when errors are detected

12.8.40 DDR SDRAM Memory Error Detect Register (MnERR_DETECT)

ERR_DETECT DDR SDRAM Memory Error Detect Register Offset 0x0E40

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MME	—														
Type	W1C	R														
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**
	—							APE	ACE	—			MBE	SBE	—	MSE
Type	R							W1C	W1C	R			W1C		R	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: W1C - Write 0b1 to clear the bit, writing 0b0 as no effect.

ERR_DETECT stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, calibration error, and memory select errors. Each bit is cleared when software writes a value of 1 to it. 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.

Table 12-61. ERR_DETECT Bit Descriptions

Bit	Reset	Description	Settings
MME 31	0	**Multiple Memory Errors** Indicates whether multiple memory errors of the same type were detected. This bit is cleared by software writing a 1 to it.	0 Multiple memory errors were not detected. 1 Multiple memory errors of the same type were detected.
— 30–9	0	Reserved. Write to zero for future compatibility.	
APE 8	0	Address parity error. Indicates whether an address parity error was detected. This bit is cleared by software writing a 1 to it.	0 An address parity error has not been detected. 1 An address parity error has been detected
ACE 7	0	**Automatic Calibration Error** Indicates whether an automatic calibration error was detected. This bit is cleared by software writing a 1 to it.	0 Automatic calibration Error has not been detected. 1 Automatic calibration Error has been detected..
— 6–4	0	Reserved. Write to zero for future compatibility.	
MBE 3	0	**Multiple-Bit Error** Indicates whether a multiple-bit error was detected. This bit is cleared by software writing a 1 to it.	0 Multiple-bit error not detected. 1 Multiple-bit error detected.
SBE 2		**Single-Bit ECC Error** Indicates whether the number of single-bit ECC errors detected is equal to the threshold set in ERR_SBE[SBET]. This bit is cleared by software writing a 1 to it.	0 The number of errors is less than the threshold. 1 The number of errors is equal or greater than the threshold.
— 1	0	Reserved. Write to zero for future compatibility.	
MSE 0	0	**Memory Select Error** Indicates whether a memory select error has been detected. This bit is cleared by software writing a 1 to it.	0 Memory select error not detected. 1 Memory select error detected.

12.8.41 DDR SDRAM Memory Error Disable Register (MnERR_DISABLE)

ERR_DISABLE DDR SDRAM Memory Error Disable Register Offset 0x0E44

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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**
	—							APED	ACED	—			MBED	SBED	—	MSED
Type	R							R/W	R/W	R			R/W		R	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_DISABLE selectively disables the DDR controller error detection circuitry. Disabled errors are not detected or reported.

Table 12-62. ERR_DISABLE Bit Descriptions

Bit	Reset	Description	Settings
— 31–9	0	Reserved. Write to zero for future compatibility.	
APED 8	0	**Address parity error disable** Address parity errors are detected if DDR_SDRAM_CFG_2[AP_EN] is set. They are reported if ERR_INT_EN[APEE] is set.	0 Address parity errors are detected. 1 Address parity errors are not detected or reported.
ACED 7	0	**Automatic Calibration Error Disable** Enables/disables automatic calibration errors detection	0 Automatic Calibration errors enabled. 1 Automatic Calibration errors disabled.
— 6–4	0	Reserved. Write to zero for future compatibility.	
MBED 3	0	**Multiple-Bit ECC Error Disable** Enables/disables multiple-bit ECC errors detection. When this bit is cleared, multiple-bit errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set.	0 Multiple-bit ECC errors detected 1 Multiple-bit ECC errors not detected or reported.
SBED 2	0	**Single-Bit ECC Error Disable** Enables/disables single-bit ECC errors detection.	0 Single-bit ECC errors detection is enabled. 1 Single-bit ECC errors detection is disabled.
1	0	Reserved. Write to zero for future compatibility.	
MSED 0	0	**Memory Select Error Disable** Enables/disables memory select errors detection.	0 Memory select errors detection is enabled. 1 Memory select errors detection is disabled.

12.8.42 DDR SDRAM Memory Error Interrupt Enable Register (MnERR_INT_EN)

ERR_INT_EN DDR SDRAM Memory Error Interrupt Enable Register Offset 0x0E48

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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**
	—							APEE	ACEE	—			MBEE	SBEE	—	MSEE
Type	R							R/W	R/W	R			R/W		R	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_INT_EN enables ECC interrupts, calibration error, address/command parity error or memory select error interrupts. When an enabled interrupt condition occurs, the DDR Controller interrupt request signal is asserted to the embedded programmable interrupt controller (EPIC)

Table 12-63. ERR_INT_EN Bit Descriptions

Bit	Reset	Description	Settings
— 31–9	0	Reserved. Write to zero for future compatibility.	
APEE 8	0	Address parity error interrupt enable	0 Address parity errors cannot generate interrupts. 1 Address parity errors generate interrupts.
ACEE 7	0	**Automatic Calibration Error Interrupt Enable** Specifies whether automatic calibration errors generate interrupts.	0 Calibration errors cannot generate interrupts. 1 Calibration errors generate interrupts.
— 6–4	0	Reserved. Write to zero for future compatibility.	
MBEE 3	0	**Multiple-Bit ECC Error Interrupt Enable** Specifies whether multiple-bit ECC errors generate interrupts.	0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
SBEE 2	0	**Single-Bit ECC Error Interrupt Enable** Specifies whether single-bit ECC errors generate interrupts.	0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
— 1	0	Reserved. Write to zero for future compatibility.	
MSEE 0		**Memory Select Error Interrupt Enable** Specifies whether memory select errors generate interrupts.	0 Memory select errors do not generate interrupts. 1 Memory select errors generate interrupts.

12.8.43 DDR SDRAM Memory Error Attributes Capture Register (MnCAPTURE_ATTRIBUTES)

CAPTURE_ATTRIBUTES DDR SDRAM Memory Error Attributes Capture Register — Offset 0x0E4C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	BNUM			—	TSIZ			—							
Type	R	R/W			R	R/W			R							
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**
	—		TTYP		—											VLD
Type	R		R/W		R											R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAPTURE_ATTRIBUTES sets attributes for errors including type, size, source, and so on.

Table 12-64. CAPTURE_ATTRIBUTES Bit Descriptions

Bit	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
BNUM 30–28	0	**Data Beat Number** Captures the data beat number for the detected error. This bit is relevant only for ECC errors.	
— 27	0	Reserved. Write to zero for future compatibility.	
TSIZ 26–24	0	**Transaction Size for Error** Captures the transaction size in 64-bit increments.	000 4 × 64 bits. 001 1 × 64 bits. 010 2 × 64 bits. 011 3 × 64 bits. All other values are reserved
— 23–14	0	Reserved. Write to zero for future compatibility.	
TTYP 13–12	0	**Transaction Type for Error** Specifies the access type that generates the error.	00 Reserved. 01 Write. 10 Read. 11 Read-modify-write.
— 11–1	0	Reserved. Write to zero for future compatibility.	
VLD 0	0	**Valid** Set as soon as valid information is captured in the error capture registers.	0 No valid information captured 1 Valid information captured

12.8.44 DDR SDRAM Memory Error Address Capture Register (MnCAPTURE_ADDRESS)

CAPTURE_ADDRESS DDR SDRAM Memory Error Address Capture Register — Offset 0x0E50

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

CAPTURE_ADDRESS holds the 32-bit of the transaction address when a DDR ECC error is detected.

Table 12-65. CAPTURE_ADDRESS Bit Descriptions

Bit	Reset	Description	Settings
CADDR 31–0	0	**Captured Address** Captures the 32 bits of the transaction address when an error is detected.	

12.8.45 DDR SDRAM Single-Bit ECC Memory Error Management Register (MnERR_SBE)

ERR_SBE DDR SDRAM Single-Bit ECC Memory Error Management Register — Offset 0x0E58

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								SBET							
Type	R								R/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**
	—								SBEC							
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ERR_SBE 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.

Table 12-66. ERR_SBE Bit Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
SBET 23–16	0	**Single-Bit Error Threshold** Establishes the number of single-bit errors that must be detected before an error condition is reported.
— 15–8	0	Reserved. Write to zero for future compatibility.
SBEC 7–0	0	**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 when this value equals the SBET value. SBEC is automatically cleared when the threshold value is reached.

12.8.46 Debug Register 2 (MnDEBUG_2)

DEBUG_2 Debug Register 2 Offset 0x0F04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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**
	—														IDLE	—
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

DEBUG_2 is used to indicate whether the controller is active.

Table 12-67. ERR_SBE Bit Descriptions

Bit	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
IDLE 1	1	**Idle** The DDR controller is in idle state.	0 Active 1 Idle
— 0	0	Reserved. Write to zero for future compatibility.	

Interrupt Handling 13

The MSC8156E interrupt system is optimized for a multi-processing environment and performs the following functions:

- Routes each of the interrupt sources to each of the extended SC3850 cores thus allowing:
  - Flexible resource allocations as well as for a symmetrical or a non-symmetrical application architecture.
  - Provides a core-to-core full interrupt mesh.
  - Provides external host-to-core signaling mechanism by virtual interrupt generation.
- Allows for the enabling/disabling of each interrupt source per core.

The MSC8156E supports both internal and external interrupt sources as well as allowing for the generation of an interrupt to external devices.

There are three device level interrupt handlers in the MSC8156E:

1. *Global interrupt controller*. Allows for the generation of virtual interrupt requests (VIRQ) as well as virtual non-maskable interrupts (VNMI) towards the cores as well as generates interrupts to external devices.
2. *General configuration block*. Concentrates and routes rare and debug interrupts to the SC3400 cores.
3. *Embedded programmable interrupt controller (EPIC)*. Concentrates all the interrupt directed at the associated core and dispatches the highest priority interrupt to the SC3400 core. Although there are various interrupts in the system designated as non-maskable interrupts (NMIs), you must program them to be non-maskable in the EPIC. This is typically done by the boot program.

Note: See the *SC3850 DSP Core Subsystem Reference Manual* for details about the EPIC. The manual is available with a signed non-disclosure agreement. Contact your local Freescale dealer or sales representative for more information.

Note: The QUICC Engine module also includes an interrupt controller that handles interrupts within the module for the dual-RISC processor system. For details about that interrupt controller and how to program it, refer to **Chapter 18**, *QUICC Engine Subsystem* and the *QUICC Engine™ Block Reference Manual with Protocol Interworking* (QEIWRM).

13.1 Global Interrupt Controller (GIC)

The GIC generates 16 VIRQs and 8 VNMIs to the DSP cores by writing to registers in the GIC memory map. The GIC also uses two additional VIRQ slots to generate a non-maskable interrupt $\overline{\text{NMI\_OUT}}$ and a maskable interrupt $\overline{\text{INT\_OUT}}$ to external devices. A virtual interrupt/VNMI is generated via a write access to the Virtual Interrupt Generation Register (VIGR) by one of the SC3400 cores or an external host CPU. **Table 13-1** describes the destination of the supported VIRQs.

Table 13-1. VIRQ Description

VIRQ Num	Destination
VIRQ_0	Connected to Virtual Interrupt 0 at SC3400
VIRQ_1	Connected to Virtual Interrupt 1 at SC3400
VIRQ_2	Connected to Virtual Interrupt 2 at SC3400
VIRQ_3	Connected to Virtual Interrupt 3 at SC3400
VIRQ_4	Connected to Virtual Interrupt 4 at SC3400
VIRQ_5	Connected to Virtual Interrupt 5 at SC3400
VIRQ_6	Connected to Virtual Interrupt 6 at SC3400
VIRQ_7	Connected to Virtual Interrupt 7 at SC3400
VIRQ_8	Connected to Virtual Interrupt 8 at SC3400
VIRQ_9	Connected to Virtual Interrupt 9 at SC3400
VIRQ_10	Connected to Virtual Interrupt 10 at SC3400
VIRQ_11	Connected to Virtual Interrupt 11 at SC3400
VIRQ_12	Connected to Virtual Interrupt 12 at SC3400
VIRQ_13	Connected to Virtual Interrupt 13 at SC3400
VIRQ_14	Connected to Virtual Interrupt 14 at SC3400
VIRQ_15	Connected to Virtual Interrupt 15 at SC3400
VIRQ_16	Connected to VNMI_0
VIRQ_17	Connected to VNMI_1
VIRQ_18	Connected to VNMI_2
VIRQ_19	Connected to VNMI_3
VIRQ_20	Connected to VNMI_4
VIRQ_21	Connected to VNMI_5
VIRQ_22	Connected to VNMI_6
VIRQ_23	Connected to VNMI_7
VIRQ_24	Use to generate $\overline{\text{INT\_OUT}}$ (see **13.2.2**, *External Interrupts*)
VIRQ_25	Use to generate $\overline{\text{NMI\_OUT}}$ (see **13.2.2**, *External Interrupts*)

The GIC has a status register to indicate whether a virtual interrupt was generated at least once, while not preventing the generation of another interrupt. The core that services the interrupt may clear this status bit by writing a value of one to it, or it may ignore this bit and work locally.

13.2 General Configuration Block

The general configuration block performs services for rare and debug interrupts generated throughout the MSC8156E before they reach the SC3850 EPICs. These services include:

- Generating ORed interrupt signals towards the SC3400 cores (see **Section 13.2.1**).
- Providing an interrupt enable bit for each interrupt source for each SC3400 core (see **Section 13.5.2**, *General Interrupt Configuration*, on page 13-26).
- Providing a status bit for each interrupt source. These bits are shared for all the SC3400 cores (see **Section 13.5.2**, *General Interrupt Configuration*, on page 13-26).

13.2.1 Interrupt Groups

The general configuration block generates 5 interrupts based on the groups of ORed interrupts described in **Table 13-2**.

Table 13-2. General Configuration Block Interrupt Sources

TDM	Debug	General	Watch Dog Timer	MAPLE-B
TDM 0 Rx error	CLASS 0 overrun	Parity error from TDM[0–3]	Watch Dog Timer 0	MAPLE general error
TDM 0 Tx error	CLASS 0 watchpoint	QUICC Engine module DRAM double soft error	Watch Dog Timer 1	
TDM 1 Rx error	CLASS 0 error	QUICC Engine module IMEM double soft error	Watch Dog Timer 2	
TDM 1 Tx error	Performance Monitor all	DMA error	Watch Dog Timer 3	
TDM 2 Rx error	Core subsystems 0 and 1 MEX address error.	DDR1 interrupt	Watch Dog Timer 4	
TDM 2 Tx error	Core subsystems 2 and 3 MEX address error.	DDR2 interrupt	Watch Dog Timer 5	
TDM 3 Rx error	Core subsystems 4 and 5 MEX address error.	OCN0 to MBus	Watch Dog Timer 6	
TDM 3 Tx error		OCN1 to MBus	Watch Dog Timer 7	
		MAPLE FFT double soft error		
		MAPLE DFT double soft error		
		MAPLE IMEM double soft error		
		MAPLE DRAM double soft error		

13.2.2 External Interrupts

The MSC8156E allows a number of external interrupt inputs to be multiplexed with the GPIO signals to enable external devices to interrupt the cores (see **Chapter 22**, *GPIO*). There are also dedicated external interrupt pins.

Note: All external $\overline{\text{IRQ}}$ signals are multiplexed options of the associated GPIO ports.

Table 13-3 summarizes all the external interrupt inputs in the MSC8156E.

Table 13-3. MSC8156E External Interrupt Pins

Name	GPIO	Direction
$\overline{\text{NMI}}$	N/A	In
$\overline{\text{NMI\_OUT}}$	N/A	Out
$\overline{\text{INT\_OUT}}$	N/A	Out
$\overline{\text{IRQ0}}$	GPIO0	In
$\overline{\text{IRQ1}}$	GPIO1	In
$\overline{\text{IRQ2}}$	GPIO2	In
$\overline{\text{IRQ3}}$	GPIO3	In
$\overline{\text{IRQ4}}$	GPIO4	In
$\overline{\text{IRQ5}}$	GPIO5	In
$\overline{\text{IRQ6}}$	GPIO6	In
$\overline{\text{IRQ7}}$	GPIO7	In
$\overline{\text{IRQ8}}$	GPIO8	In
$\overline{\text{IRQ9}}$	GPIO9	In
$\overline{\text{IRQ10}}$	GPIO10	In
$\overline{\text{IRQ11}}$	GPIO11	In
$\overline{\text{IRQ12}}$	GPIO12	In
$\overline{\text{IRQ13}}$	GPIO13	In
$\overline{\text{IRQ14}}$	GPIO14	In
$\overline{\text{IRQ15}}$	GPIO15	In

$\overline{\text{INT\_OUT}}$ is asserted when VIRQ_24 is asserted. $\overline{\text{NMI\_OUT}}$ is asserted when VIRQ_25 is asserted.

13.2.3 Interrupt Handling

The MSC8156E interrupts sources can be grouped in to four basic types:

1. Interrupts that represent a single interrupt source and are routed directly to the cores (for example, the TDM0 Rx first threshold interrupt).
2. Interrupts that represent multiple interrupt sources and are routed directly to the cores (for example, all I^2C interrupts).
3. Interrupts that represent a single interrupt source and are routed to the cores via the general configuration block (for example, MAPLE general error interrupt).
4. Interrupts that represent multiple interrupt sources and are routed to the cores via the General Configuration Block (for example, parity error from TDM[0–3] interrupt).

Figure 13-1 outlines the flow for handling the various types of interrupts.

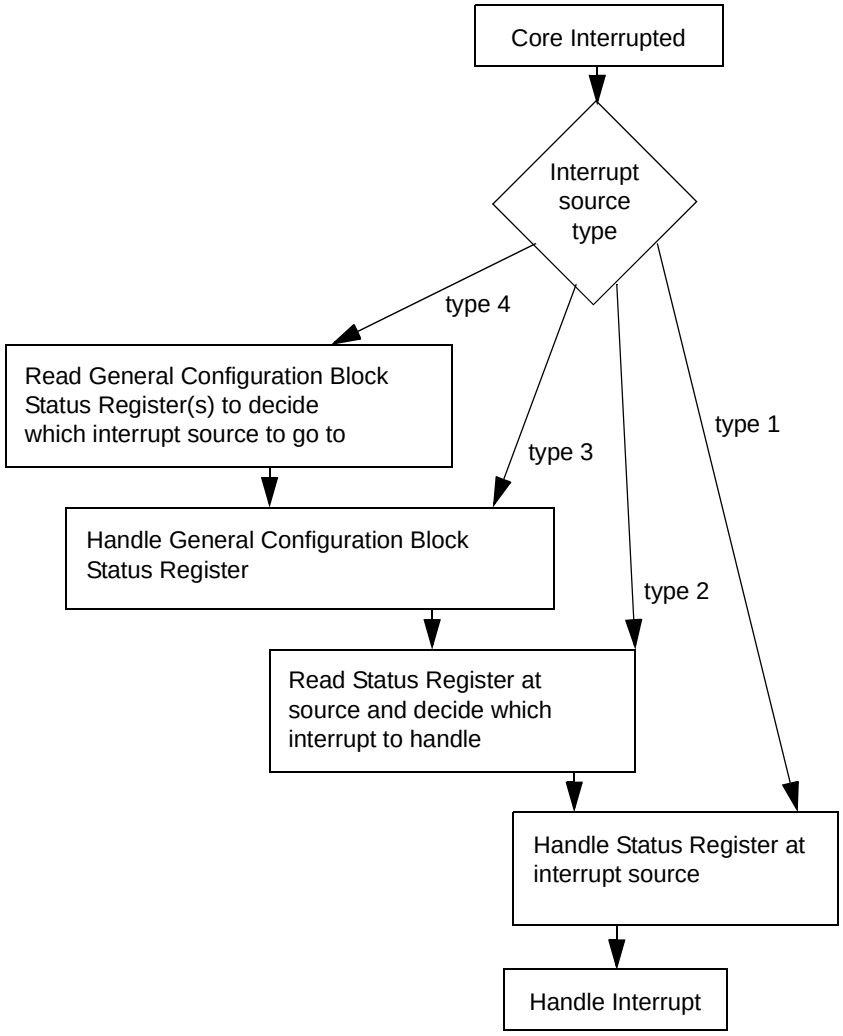


Figure 13-1. Interrupt Handling Flow

As an example of handling a type 4 interrupt, processing of a QUICC Engine interrupt is provided. In **Table 13-2**, there are two types of QUICC Engine interrupts that are routed through the GCRs:

- DRAM double soft error
- IMEM double soft error

When enabled and unmasked, both interrupts are concentrated to the GCR general interrupt which is routed to all cores. The GIER1_[0–5] registers allow the user to mask/unmask the interrupts for any or all of the cores.

If the general interrupt from the GCR is received by a core (index 245), the ISR should read GIR1 (GCR offset 0x80) to identify the origin of the general interrupt. Bits 18 and 19 in the register represent DRAM double soft error and IMEM double soft error, respectively. After identifying that the interrupt is coming from the QUICC Engine subsystem, the cores can handle the interrupt in the same way as other QUICC Engine interrupts that are directed to the DSP core subsystems.

There are nine other QUICC Engine interrupts that use indexes 132–141 as indicated in **Table 13-4**. Details about the QUICC Engine interrupt controller and how to use this interrupts is provided in **Chapter 18**, *QUICC Engine Subsystem* and the *QUICC Engine™ Block Reference Manual with Interworking* (QEIWRM).

13.3 Interrupt Mapping

The EPIC can support up to 256 interrupt sources that can be level-, edge-, or double-edge triggered. The interrupts can have an assigned priority from 1(lowest) to 31 (highest) as well as non-maskable (priority 32). The first 34 interrupt sources are used internally by the SC3850 DSP cores. The core-to-core interrupt mesh uses another 12 interrupts for core-to-core communication. All other interrupts are used by the MSC8156E device. The MSC8156E does not implement all of these possible sources.

Table 13-4 describes the interrupt capabilities (level/edge) and index for each interrupt source. The interrupt default priority at wake up is 0 (all interrupts are ignored). Interrupt sources routed via the general configuration block do not appear in the table because their function is set by the GCR.

Table 13-4. MSC8156E Interrupt Table

Interrupt Description	IRQ index	Level	Edge
Core Subsystem Interrupt Mesh			
From Core Subsystem 0	34	+	+
From Core Subsystem 0	35	+	+
From Core Subsystem 1	36	+	+
From Core Subsystem 1	37	+	+

Table 13-4. MSC8156E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
From Core Subsystem 2	38	+	+
From Core Subsystem 2	39	+	+
From Core Subsystem 3	40	+	+
From Core Subsystem 3	41	+	+
From Core Subsystem 4	42	+	+
From Core Subsystem 4	43	+	+
From Core Subsystem 5	44	+	+
From Core Subsystem 5	45	+	+
TDM			
TDM 0 Rx first threshold	50	+	+
TDM 0 Rx second threshold	51	+	+
TDM 0 Tx first threshold	52	+	+
TDM 0 Tx second threshold	53	+	+
TDM 1 Rx first threshold	54	+	+
TDM 1 Rx second threshold	55	+	+
TDM 1 Tx first threshold	56	+	+
TDM 1 Tx second threshold	57	+	+
TDM 2 Rx first threshold	58	+	+
TDM 2 Rx second threshold	59	+	+
TDM 2 Tx first threshold	60	+	+
TDM 2 Tx second threshold	61	+	+
TDM 3 Rx first threshold	62	+	+
TDM 3 Rx second threshold	63	+	+
TDM 3 Tx first threshold	64	+	+
TDM 3 Tx second threshold	65	+	+
Serial RapidIO			
Serial RapidIO message in 0	82	+	—
Serial RapidIO message in 1	83	+	—
Serial RapidIO message out 0	84	+	—
Serial RapidIO message out 1	85	+	—
Serial RapidIO doorbell inbound	86	+	—
Serial RapidIO doorbell outbound	87	+	—
Serial RapidIO general error	88	+	—
Ethernet 1			
Ethernet 1 all	91	+	—
Ethernet 1 Rx 0	92	+	—
Ethernet 1 Rx 1	93	+	—
Ethernet 1 Rx 2	94	+	—
Ethernet 1 Rx 3	95	+	—

Table 13-4. MSC8156E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
Ethernet 1 Rx 4	96	+	—
Ethernet 1 Rx 5	97	+	—
Ethernet 1 Rx 6	98	+	—
Ethernet 1 Rx 7	99	+	—
Ethernet 1 Tx 0	100	+	—
Ethernet 1 Tx 1	101	+	—
Ethernet 1 Tx 2	102	+	—
Ethernet 1 Tx 3	103	+	—
Ethernet 1 Tx 4	104	+	—
Ethernet 1 Tx 5	105	+	—
Ethernet 1 Tx 6	106	+	—
Ethernet 1 Tx 7	107	+	—
Ethernet 2			
Ethernet 2 all	109	+	—
Ethernet 2 Rx 0	110	+	—
Ethernet 2 Rx 1	111	+	—
Ethernet 2 Rx 2	112	+	—
Ethernet 2 Rx 3	113	+	—
Ethernet 2 Rx 4	114	+	—
Ethernet 2 Rx 5	115	+	—
Ethernet 2 Rx 6	116	+	—
Ethernet 2 Rx 7	117	+	—
Ethernet 2 Tx 0	118	+	—
Ethernet 2 Tx 1	119	+	—
Ethernet 2 Tx 2	120	+	—
Ethernet 2 Tx 3	121	+	—
Ethernet 2 Tx 4	122	+	—
Ethernet 2 Tx 5	123	+	—
Ethernet 2 Tx 6	124	+	—
Ethernet 2 Tx 7	125	+	—
PCI Express			
PCI Express INTA	127	+	—
PCI Express INTB	128	+	—
PCI Express INTC	129	+	—
PCI Express INTD	130	+	—
PCI Express general interrupt	131	+	—
QUICC Engine Subsystem			
QUICC Engine interrupt output 0	132	+	—
QUICC Engine interrupt output 1	133	+	—

Table 13-4. MSC8156E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
QUICC Engine interrupt output 2	134	+	—
QUICC Engine interrupt output 3	135	+	—
QUICC Engine interrupt output 4	136	+	—
QUICC Engine interrupt output 5	137	+	—
QUICC Engine interrupt output 6	138	+	—
QUICC Engine interrupt output 7	139	+	—
QUICC Engine module critical	140	+	—
QUICC Engine module regular	141	+	—
DMA			
DMA channel 0 EOB	144	+	—
DMA channel 1 EOB	145	+	—
DMA channel 2 EOB	146	+	—
DMA channel 3 EOB	147	+	—
DMA channel 4 EOB	148	+	—
DMA channel 5 EOB	149	+	—
DMA channel 6 EOB	150	+	—
DMA channel 7 EOB	151	+	—
DMA channel 8 EOB	152	+	—
DMA channel 9 EOB	153	+	—
DMA channel 10 EOB	154	+	—
DMA channel 11 EOB	155	+	—
DMA channel 12 EOB	156	+	—
DMA channel 13 EOB	157	+	—
DMA channel 14 EOB	158	+	—
DMA channel 15 EOB	159	+	—
Timer 0			
Timer 0 Channel 0	160	+	—
Timer 0 Channel 1	161	+	—
Timer 0 Channel 2	162	+	—
Timer 0 Channel 3	163	+	—
Timer 1			
Timer 1 Channel 0	164	+	—
Timer 1 Channel 1	165	+	—
Timer 1 Channel 2	166	+	—
Timer 1 Channel 3	167	+	—
Timer 2			
Timer 2 Channel 0	168	+	—
Timer 2 Channel 1	169	+	—
Timer 2 Channel 2	170	+	—

Table 13-4. MSC8156E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
Timer 2 Channel 3	171	+	—
Timer 3			
Timer 3 Channel 0	172	+	—
Timer 3 Channel 1	173	+	—
Timer 3 Channel 2	174	+	—
Timer 3 Channel 3	175	+	—
UART			
UART all	176	+	—
Global Interrupt Controller			
Virtual Interrupt 0	177	—	+
Virtual Interrupt 1	178	—	+
Virtual Interrupt 2	179	—	+
Virtual Interrupt 3	180	—	+
Virtual Interrupt 4	181	—	+
Virtual Interrupt 5	182	—	+
Virtual Interrupt 6	183	—	+
Virtual Interrupt 7	184	—	+
Virtual Interrupt 8	185	—	+
Virtual Interrupt 9	186	—	+
Virtual Interrupt 10	187	—	+
Virtual Interrupt 11	188	—	+
Virtual Interrupt 12	189	—	+
Virtual Interrupt 13	190	—	+
Virtual Interrupt 14	191	—	+
Virtual Interrupt 15	192	—	+
Virtual Non Maskable Interrupt 0	193	—	+
Virtual Non Maskable Interrupt 1	194	—	+
Virtual Non Maskable Interrupt 2	195	—	+
Virtual Non Maskable Interrupt 3	196	—	+
Virtual Non Maskable Interrupt 4	197	—	+
Virtual Non Maskable Interrupt 5	198	—	+
Virtual Non Maskable Interrupt 6	199	—	+
Virtual Non Maskable Interrupt 7	200	—	+
SEC			
Primary interrupt	201	+	—
Secondary interrupt	202	+	—
OCNDMA0			
Channel 0 Interrupt	203	+	—
Channel 1 Interrupt	204	+	—

Table 13-4. MSC8156E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
Channel 2 Interrupt	205	+	—
Channel 3 Interrupt	206	+	—
I^2C			
I^2C all	208	+	—
MAPLE-B			
MAPLE BD 0	209	+	+
MAPLE BD 1	210	+	+
MAPLE BD 2	211	+	+
MAPLE BD 3	212	+	+
MAPLE BD 4	213	+	+
MAPLE BD 5	214	+	+
MAPLE BD 6	215	+	+
MAPLE BD 7	216	+	+
MAPLE BD 8	217	+	+
MAPLE BD 9	218	+	+
MAPLE BD 10	219	+	+
MAPLE BD 11	220	+	+
MAPLE BD 12	221	+	+
MAPLE BD 13	222	+	+
MAPLE BD 14	223	+	+
MAPLE BD 15	224	+	+
External $\overline{\text{IRQ}}$s			
$\overline{\text{IRQ0}}$ (see note)	226	+	+
$\overline{\text{IRQ1}}$ (see note)	227	+	+
$\overline{\text{IRQ2}}$ (see note)	228	+	+
$\overline{\text{IRQ3}}$ (see note)	229	+	+
$\overline{\text{IRQ4}}$ (see note)	230	+	+
$\overline{\text{IRQ5}}$ (see note)	231	+	+
$\overline{\text{IRQ6}}$ (see note)	232	+	+
$\overline{\text{IRQ7}}$ (see note)	233	+	+
$\overline{\text{IRQ8}}$ (see note)	234	+	+
$\overline{\text{IRQ9}}$ (see note)	235	+	+
$\overline{\text{IRQ10}}$ (see note)	236	+	+
$\overline{\text{IRQ11}}$ (see note)	237	+	+
$\overline{\text{IRQ12}}$ (see note)	238	+	+
$\overline{\text{IRQ13}}$ (see note)	239	+	+
$\overline{\text{IRQ14}}$ (see note)	240	+	+
$\overline{\text{IRQ15}}$ (see note)	241	+	+

Table 13-4. MSC8156E Interrupt Table (Continued)

Interrupt Description	IRQ index	Level	Edge
NMI (see note)	242	+	+
General Configuration Block			
ORed TDM interrupts	243	+	—
ORed Debug Interrupts	244	+	—
ORed General Interrupts	245	+	—
ORed Watch Dog Timer Interrupts	246	+	—
ORed MAPLE Interrupts	247	+	—
OCNDMA1			
Channel 0 Interrupt	248	+	—
Channel 1 Interrupt	249	+	—
Channel 2 Interrupt	250	+	—
Channel 3 Interrupt	251	+	—

Note: For NMIs and IRQs, when configured as edge-triggered interrupts, assertion of the interrupt is sensed by the cores when the signals are changing state from 1 to 0.

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
Trap0	Internal exception (generated by a TRAP0 instruction)	0	0x0	—	—	0	0x0	1	16 B
Trap1	Internal exception (generated by a TRAP1 instruction)			—	—	16	0x10	1	16 B
Trap2	Internal exception (generated by a TRAP2 instruction)			—	—	32	0x20	1	16 B
Trap3	Internal exception (generated by a TRAP3 instruction)			—	—	48	0x30	1	16 B
Reserved	—	1	0x1	—	—	64	0x40	4	64 B
ILLEGAL	Illegal instruction or set.	2	0x2	—	—	128	0x80	4	64 B
DEBUG	• Debug exception (OCE) • DEBUG/EV instruction and EDCA are Precise After • EDCD is Precise After (+2 read, +5 write)	3	0x3	—	—	192	0xC0	4	64 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
OVERFLOW	Overflow exception (DALU).	4	0x4	—	—	256	0x100	4	64 B
Reserved	—	5	0x5	—	—	320	0x140	1	16 B
OCE	OCE exception.			—	—	328	0x148	1	16 B
IMMUAE	Instruction MMU address error.			—	—	336	0x150	1	16 B
DMMUAE	Data MMU address error.			—	—	344	0x158	1	16 B
Reserved	—			—	—	352	0x160	1	16 B
IEDC	Instruction EDC error.			—	—	360	0x168	1	16 B
DEDC	Data EDC error.			—	—	368	0x170	1	16 B
Reserved	—			—	—	376	0x178	1	16 B
Reserved	—	6	0x6	—	—	384	0x180	4	64 B
Reserved	—	7	0x7	—	—	448	0x1C0	4	64 B
I_MIFER	Master interface errors from the MMU (NMI)	8	0x8	0	0x0	512	0x200	1	16 B
I_SIFER	Slave interface errors from the MMU (NMI)	9	0x9	1	0x1	528	0x210	1	16 B
I_WBBEDC	WBB soft data error (NMI)	10	0xA	2	0x2	544	0x220	1	16 B
Reserved	Reserved	11	0xB	3	0x3	560	0x230	1	16 B
I_ICDME	ICache double match error (NMI)	12	0xC	4	0x4	576	0x240	1	16 B
I_DCDME	DCache double match error (NMI)	13	0xD	5	0x5	592	0x250	1	16 B
I_L2NM2	L2 non-mapped M2 access (NMI)	14	0xE	6	0x6	608	0x260	1	16 B
I_L2NAE	L2 non-aligned non-allocated access (NMI)	15	0xF	7	0x7	624	0x270	1	16 B
Reserved	Reserved for internal DSP subsystem use	16	0x10	8	0x8	640	0x280	1	16 B
I_ICAES	ICache end-of-sweep operation exception	17	0x11	9	0x9	656	0x290	1	16 B
I_DCAES	DCache end-of-sweep operation exception	18	0x12	10	0xA	672	0x2A0	1	16 B
I_L2AES	L2 Cache end-of-sweep operation exception operational in a DSP subsystem with L2 cache	19	0x13	11	0xB	688	0x2B0	1	16 B
I_TM0	Timer 0 interrupt	20	0x14	12	0xC	704	0x2C0	1	16 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
I_TM1	Timer 1 interrupt	21	0x15	13	0xD	720	0x2D0	1	16 B
I_DPUA	DPU interrupt A	22	0x16	14	0xE	736	0x2E0	1	16 B
i_DPUB	DPU interrupt B	23	0x17	15	0xF	752	0x2F0	1	16 B
I_ICNCH	ICache noncacheable hit exception	24	0x18	16	0x10	768	0x300	1	16 B
I_DCNCH	DCache noncacheable hit exception	25	0x19	17	0x11	784	0x310	1	16 B
I_L2NCH	L2 cache noncacheable hit exception	26	0x1A	18	0x12	800	0x320	1	16 B
I_L2ESP	L2 cache end-of-software prefetch	27	0x1B	19	0x13	816	0x330	1	16 B
Reserved	—	28	0x1C	20	0x14	832	0x340	1	16 B
Reserved	—	29	0x1D	21	0x15	848	0x350	1	16 B
Reserved	—	30	0x1E	22	0x16	864	0x360	1	16 B
Reserved	—	31	0x1F	23	0x17	880	0x370	1	16 B
Reserved	—	32	0x20	24	0x18	896	0x380	1	16 B
Reserved	—	33	0x21	25	0x19	912	0x390	1	16 B
Reserved	—	34	0x22	26	0x1A	928	0x3A0	1	16 B
Reserved	—	35	0x23	27	0x1B	944	0x3B0	1	16 B
Reserved	—	36	0x24	28	0x1C	960	0x3C0	1	16 B
Reserved	—	37	0x25	29	0x1D	976	0x3D0	1	16 B
Reserved	—	38	0x26	30	0x1E	992	0x3E0	1	16 B
Reserved	—	39	0x27	31	0x1F	1008	0x3F0	1	16 B
Reserved	—	40	0x28	32	0x20	1024	0x400	1	16 B
Reserved	—	41	0x29	33	0x21	1040	0x410	1	16 B
IRQ34	From Core Subsystem 0	42	0x2A	34	0x22	1056	0x420	1	16 B
IRQ35	From Core Subsystem 0	43	0x2B	35	0x23	1072	0x430	1	16 B
IRQ36	From Core Subsystem 1	44	0x2C	36	0x24	1088	0x440	1	16 B
IRQ37	From Core Subsystem 1	45	0x2D	37	0x25	1104	0x450	1	16 B
IRQ38	From Core Subsystem 2	46	0x2E	38	0x26	1120	0x460	1	16 B
IRQ39	From Core Subsystem 2	47	0x2F	39	0x27	1136	0x470	1	16 B
IRQ40	From Core Subsystem 3	48	0x30	40	0x28	1152	0x480	1	16 B
IRQ41	From Core Subsystem 3	49	0x31	41	0x29	1168	0x490	1	16 B
IRQ42	—	50	0x32	42	0x2A	1184	0x4A0	1	16 B
IRQ43	—	51	0x33	43	0x2B	1200	0x4B0	1	16 B
IRQ44	—	52	0x34	44	0x2C	1216	0x4C0	1	16 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ45	—	53	0x35	45	0x2D	1232	0x4D0	1	16 B
IRQ46	—	54	0x36	46	0x2E	1248	0x4E0	1	16 B
IRQ47	—	55	0x37	47	0x2F	1264	0x4F0	1	16 B
IRQ48	—	56	0x38	48	0x30	1280	0x500	1	16 B
IRQ49	—	57	0x39	49	0x31	1296	0x510	1	16 B
IRQ50	TDM 0 Rx first threshold	58	0x3A	50	0x32	1312	0x520	1	16 B
IRQ51	TDM 0 Rx second threshold	59	0x3B	51	0x33	1328	0x530	1	16 B
IRQ52	TDM 0 Tx first threshold	60	0x3C	52	0x34	1344	0x540	1	16 B
IRQ53	TDM 0 Tx second threshold	61	0x3D	53	0x35	1360	0x550	1	16 B
IRQ54	TDM 1 Rx first threshold	62	0x3E	54	0x36	1376	0x560	1	16 B
IRQ55	TDM 1 Rx second threshold	63	0x3F	55	0x37	1392	0x570	1	16 B
IRQ56	TDM 1 Tx first threshold	64	0x40	56	0x38	1408	0x580	1	16 B
IRQ57	TDM 1 Tx second threshold	65	0x41	57	0x39	1424	0x590	1	16 B
IRQ58	TDM 2 Rx first threshold	66	0x42	58	0x3A	1440	0x5A0	1	16 B
IRQ59	TDM 2 Rx second threshold	67	0x43	59	0x3B	1456	0x5B0	1	16 B
IRQ60	TDM 2 Tx first threshold	68	0x44	60	0x3C	1472	0x5C0	1	16 B
IRQ61	TDM 2 Tx second threshold	69	0x45	61	0x3D	1488	0x5D0	1	16 B
IRQ62	TDM 3 Rx first threshold	70	0x46	62	0x3E	1504	0x5E0	1	16 B
IRQ63	TDM 3 Rx second threshold	71	0x47	63	0x3F	1520	0x5F0	1	16 B
IRQ64	TDM 3 Tx first threshold	72	0x48	64	0x40	1536	0x600	1	16 B
IRQ65	TDM 3 Tx second threshold	73	0x49	65	0x41	1552	0x610	1	16 B
IRQ66	—	74	0x4A	66	0x42	1568	0x620	1	16 B
IRQ67	—	75	0x4B	67	0x43	1584	0x630	1	16 B
IRQ68	—	76	0x4C	68	0x44	1600	0x640	1	16 B
IRQ69	—	77	0x4D	69	0x45	1616	0x650	1	16 B
IRQ70	—	78	0x4E	70	0x46	1632	0x660	1	16 B
IRQ71	—	79	0x4F	71	0x47	1648	0x670	1	16 B
IRQ72	—	80	0x50	72	0x48	1664	0x680	1	16 B
IRQ73	—	81	0x51	73	0x49	1680	0x690	1	16 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ74	—	82	0x52	74	0x4A	1696	0x6A0	1	16 B
IRQ75	—	83	0x53	75	0x4B	1712	0x6B0	1	16 B
IRQ76	—	84	0x54	76	0x4C	1728	0x6C0	1	16 B
IRQ77	—	85	0x55	77	0x4D	1744	0x6D0	1	16 B
IRQ78	—	86	0x56	78	0x4E	1760	0x6E0	1	16 B
IRQ79	—	87	0x57	79	0x4F	1776	0x6F0	1	16 B
IRQ80	—	88	0x58	80	0x50	1792	0x700	1	16 B
IRQ81	—	89	0x59	81	0x51	1808	0x710	1	16 B
IRQ82	Serial RapidIO message in 0	90	0x5A	82	0x52	1824	0x720	1	16 B
IRQ83	Serial RapidIO message in 1	91	0x5B	83	0x53	1840	0x730	1	16 B
IRQ84	Serial RapidIO message out 0	92	0x5C	84	0x54	1856	0x740	1	16 B
IRQ85	Serial RapidIO message out 1	93	0x5D	85	0x55	1872	0x750	1	16 B
IRQ86	Serial RapidIO doorbell inbound	94	0x5E	86	0x56	1888	0x760	1	16 B
IRQ87	Serial RapidIO doorbell outbound	95	0x5F	87	0x57	1904	0x770	1	16 B
IRQ88	Serial RapidIO general error	96	0x60	88	0x58	1920	0x780	1	16 B
IRQ89	—	97	0x61	89	0x59	1936	0x790	1	16 B
IRQ90	—	98	0x62	90	0x5A	1952	0x7A0	1	16 B
IRQ91	Ethernet 1 all	99	0x63	91	0x5B	1968	0x7B0	1	16 B
IRQ92	Ethernet 1 Rx 0	100	0x64	92	0x5C	1984	0x7C0	1	16 B
IRQ93	Ethernet 1 Rx 1	101	0x65	93	0x5D	2000	0x7D0	1	16 B
IRQ94	Ethernet 1 Rx 2	102	0x66	94	0x5E	2016	0x7E0	1	16 B
IRQ95	Ethernet 1 Rx 3	103	0x67	95	0x5F	2032	0x7F0	1	16 B
IRQ96	Ethernet 1 Rx 4	104	0x68	96	0x60	2048	0x800	1	16 B
IRQ97	Ethernet 1 Rx 5	105	0x69	97	0x61	2064	0x810	1	16 B
IRQ98	Ethernet 1 Rx 6	106	0x6A	98	0x62	2080	0x820	1	16 B
IRQ99	Ethernet 1 Rx 7	107	0x6B	99	0x63	2096	0x830	1	16 B
IRQ100	Ethernet 1 Tx 0	108	0x6C	100	0x64	2112	0x840	1	16 B
IRQ101	Ethernet 1 Tx 1	109	0x6D	101	0x65	2128	0x850	1	16 B
IRQ102	Ethernet 1 Tx 2	110	0x6E	102	0x66	2144	0x860	1	16 B
IRQ103	Ethernet 1 Tx 3	111	0x6F	103	0x67	2160	0x870	1	16 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ104	Ethernet 1 Tx 4	112	0x70	104	0x68	2176	0x880	1	16 B
IRQ105	Ethernet 1 Tx 5	113	0x71	105	0x69	2192	0x890	1	16 B
IRQ106	Ethernet 1 Tx 6	114	0x72	106	0x6A	2208	0x8A0	1	16 B
IRQ107	Ethernet 1 Tx 7	115	0x73	107	0x6B	2224	0x8B0	1	16 B
IRQ108	—	116	0x74	108	0x6C	2240	0x8C0	1	16 B
IRQ109	Ethernet 2 all	117	0x75	109	0x6D	2256	0x8D0	1	16 B
IRQ110	Ethernet 2 Rx 0	118	0x76	110	0x6E	2272	0x8E0	1	16 B
IRQ111	Ethernet 2 Rx 1	119	0x77	111	0x6F	2288	0x8F0	1	16 B
IRQ112	Ethernet 2 Rx 2	120	0x78	112	0x70	2304	0x900	1	16 B
IRQ113	Ethernet 2 Rx 3	121	0x79	113	0x71	2320	0x910	1	16 B
IRQ114	Ethernet 2 Rx 4	122	0x7A	114	0x72	2336	0x920	1	16 B
IRQ115	Ethernet 2 Rx 5	123	0x7B	115	0x73	2352	0x930	1	16 B
IRQ116	Ethernet 2 Rx 6	124	0x7C	116	0x74	2368	0x940	1	16 B
IRQ117	Ethernet 2 Rx 7	125	0x7D	117	0x75	2384	0x950	1	16 B
IRQ118	Ethernet 2 Tx 0	126	0x7E	118	0x76	2400	0x960	1	16 B
IRQ119	Ethernet 2 Tx 1	127	0x7F	119	0x77	2416	0x970	1	16 B
IRQ120	Ethernet 2 Tx 2	128	0x80	120	0x78	2432	0x980	1	16 B
IRQ121	Ethernet 2 Tx 3	129	0x81	121	0x79	2448	0x990	1	16 B
IRQ122	Ethernet 2 Tx 4	130	0x82	122	0x7A	2464	0x9A0	1	16 B
IRQ123	Ethernet 2 Tx 5	131	0x83	123	0x7B	2480	0x9B0	1	16 B
IRQ124	Ethernet 2 Tx 6	132	0x84	124	0x7C	2496	0x9C0	1	16 B
IRQ125	Ethernet 2 Tx 7	133	0x85	125	0x7D	2512	0x9D0	1	16 B
IRQ126	—	134	0x86	126	0x7E	2528	0x9E0	1	16 B
IRQ127	PCI Express INTA	135	0x87	127	0x7F	2544	0x9F0	1	16 B
IRQ128	PCI Express INTB	136	0x88	128	0x80	2560	0xA00	1	16 B
IRQ129	PCI Express INTC	137	0x89	129	0x81	2576	0xA10	1	16 B
IRQ130	PCI Express INTD	138	0x8A	130	0x82	2592	0xA20	1	16 B
IRQ131	PCI Express general interrupt	139	0x8B	131	0x83	2608	0xA30	1	16 B
IRQ132	QUICC Engine interrupt output 0	140	0x8C	132	0x84	2624	0xA40	1	16 B
IRQ133	QUICC Engine interrupt output 1	141	0x8D	133	0x85	2640	0xA50	1	16 B
IRQ134	QUICC Engine interrupt output 2	142	0x8E	134	0x86	2656	0xA60	1	16 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ135	QUICC Engine interrupt output 3	143	0x8F	135	0x87	2672	0xA70	1	16 B
IRQ136	QUICC Engine interrupt output 4	144	0x90	136	0x88	2688	0xA80	1	16 B
IRQ137	QUICC Engine interrupt output 5	145	0x91	137	0x89	2704	0xA90	1	16 B
IRQ138	QUICC Engine interrupt output 6	146	0x92	138	0x8A	2720	0xAA0	1	16 B
IRQ139	QUICC Engine interrupt output 7	147	0x93	139	0x8B	2736	0xAB0	1	16 B
IRQ140	QUICC Engine module critical	148	0x94	140	0x8C	2752	0xAC0	1	16 B
IRQ141	QUICC Engine module regular	149	0x95	141	0x8D	2768	0xAD0	1	16 B
IRQ142	—	150	0x96	142	0x8E	2784	0xAE0	1	16 B
IRQ143	—	151	0x97	143	0x8F	2800	0xAF0	1	16 B
IRQ144	DMA channel 0 EOB	152	0x98	144	0x90	2816	0xB00	1	16 B
IRQ145	DMA channel 1 EOB	153	0x99	145	0x91	2832	0xB10	1	16 B
IRQ146	DMA channel 2 EOB	154	0x9A	146	0x92	2848	0xB20	1	16 B
IRQ147	DMA channel 3 EOB	155	0x9B	147	0x93	2864	0xB30	1	16 B
IRQ148	DMA channel 4 EOB	156	0x9C	148	0x94	2880	0xB40	1	16 B
IRQ149	DMA channel 5 EOB	157	0x9D	149	0x95	2896	0xB50	1	16 B
IRQ150	DMA channel 6 EOB	158	0x9E	150	0x96	2912	0xB60	1	16 B
IRQ151	DMA channel 7 EOB	159	0x9F	151	0x97	2928	0xB70	1	16 B
IRQ152	DMA channel 8 EOB	160	0xA0	152	0x98	2944	0xB80	1	16 B
IRQ153	DMA channel 9 EOB	161	0xA1	153	0x99	2960	0xB90	1	16 B
IRQ154	DMA channel 10 EOB	162	0xA2	154	0x9A	2976	0xBA0	1	16 B
IRQ155	DMA channel 11 EOB	163	0xA3	155	0x9B	2992	0xBB0	1	16 B
IRQ156	DMA channel 12 EOB	164	0xA4	156	0x9C	3008	0xBC0	1	16 B
IRQ157	DMA channel 13 EOB	165	0xA5	157	0x9D	3024	0xBD0	1	16 B
IRQ158	DMA channel 14 EOB	166	0xA6	158	0x9E	3040	0xBE0	1	16 B
IRQ159	DMA channel 15 EOB	167	0xA7	159	0x9F	3056	0xBF0	1	16 B
IRQ160	Timer 0 Channel 0	168	0xA8	160	0xA0	3072	0xC00	1	16 B
IRQ161	Timer 0 Channel 1	169	0xA9	161	0xA1	3088	0xC10	1	16 B
IRQ162	Timer 0 Channel 2	170	0xAA	162	0xA2	3104	0xC20	1	16 B
IRQ163	Timer 0 Channel 3	171	0xAB	163	0xA3	3120	0xC30	1	16 B
IRQ164	Timer 1 Channel 0	172	0xAC	164	0xA4	3136	0xC40	1	16 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ165	Timer 1 Channel 1	173	0xAD	165	0xA5	3152	0xC50	1	16 B
IRQ166	Timer 1 Channel 2	174	0xAE	166	0xA6	3168	0xC60	1	16 B
IRQ167	Timer 1 Channel 3	175	0xAF	167	0xA7	3184	0xC70	1	16 B
IRQ168	Timer 2 Channel 0	176	0xB0	168	0xA8	3200	0xC80	1	16 B
IRQ169	Timer 2 Channel 1	177	0xB1	169	0xA9	3216	0xC90	1	16 B
IRQ170	Timer 2 Channel 2	178	0xB2	170	0xAA	3232	0xCA0	1	16 B
IRQ171	Timer 2 Channel 3	179	0xB3	171	0xAB	3248	0xCB0	1	16 B
IRQ172	Timer 3 Channel 0	180	0xB4	172	0xAC	3264	0xCC0	1	16 B
IRQ173	Timer 3 Channel 1	181	0xB5	173	0xAD	3280	0xCD0	1	16 B
IRQ174	Timer 3 Channel 2	182	0xB6	174	0xAE	3296	0xCE0	1	16 B
IRQ175	Timer 3 Channel 3	183	0xB7	175	0xAF	3312	0xCF0	1	16 B
IRQ176	UART all	184	0xB8	176	0xB0	3328	0xD00	1	16 B
IRQ177	Virtual Interrupt 0	185	0xB9	177	0xB1	3344	0xD10	1	16 B
IRQ178	Virtual Interrupt 1	186	0xBA	178	0xB2	3360	0xD20	1	16 B
IRQ179	Virtual Interrupt 2	187	0xBB	179	0xB3	3376	0xD30	1	16 B
IRQ180	Virtual Interrupt 3	188	0xBC	180	0xB4	3392	0xD40	1	16 B
IRQ181	Virtual Interrupt 4	189	0xBD	181	0xB5	3408	0xD50	1	16 B
IRQ182	Virtual Interrupt 5	190	0xBE	182	0xB6	3424	0xD60	1	16 B
IRQ183	Virtual Interrupt 6	191	0xBF	183	0xB7	3440	0xD70	1	16 B
IRQ184	Virtual Interrupt 7	192	0xC0	184	0xB8	3456	0xD80	1	16 B
IRQ185	Virtual Interrupt 8	193	0xC1	185	0xB9	3472	0xD90	1	16 B
IRQ186	Virtual Interrupt 9	194	0xC2	186	0xBA	3488	0xDA0	1	16 B
IRQ187	Virtual Interrupt 10	195	0xC3	187	0xBB	3504	0xDB0	1	16 B
IRQ188	Virtual Interrupt 11	196	0xC4	188	0xBC	3520	0xDC0	1	16 B
IRQ189	Virtual Interrupt 12	197	0xC5	189	0xBD	3536	0xDD0	1	16 B
IRQ190	Virtual Interrupt 13	198	0xC6	190	0xBE	3552	0xDE0	1	16 B
IRQ191	Virtual Interrupt 14	199	0xC7	191	0xBF	3568	0xDF0	1	16 B
IRQ192	Virtual Interrupt 15	200	0xC8	192	0xC0	3584	0xE00	0.5	8 B
IRQ193	Virtual Non Maskable Interrupt 0	201	0xC9	193	0xC1	3592	0xE08	0.5	8 B
IRQ194	Virtual Non Maskable Interrupt 1	202	0xCA	194	0xC2	3600	0xE10	0.5	8 B
IRQ195	Virtual Non Maskable Interrupt 2	203	0xCB	195	0xC3	3608	0xE18	0.5	8 B
IRQ196	Virtual Non Maskable Interrupt 3	204	0xCC	196	0xC4	3616	0xE20	0.5	8 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ197	Virtual Non Maskable Interrupt 4	205	0xCD	197	0xC5	3624	0xE28	0.5	8 B
IRQ198	Virtual Non Maskable Interrupt 5	206	0xCE	198	0xC6	3632	0xE30	0.5	8 B
IRQ199	Virtual Non Maskable Interrupt 6	207	0xCF	199	0xC7	3640	0xE38	0.5	8 B
IRQ200	Virtual Non Maskable Interrupt 7	208	0xD0	200	0xC8	3648	0xE40	0.5	8 B
IRQ201	Primary interrupt	209	0xD1	201	0xC9	3656	0xE48	0.5	8 B
IRQ202	Secondary interrupt	210	0xD2	202	0xCA	3664	0xE50	0.5	8 B
IRQ203	Channel 0 Interrupt	211	0xD3	203	0xCB	3672	0xE58	0.5	8 B
IRQ204	Channel 1 Interrupt	212	0xD4	204	0xCC	3680	0xE60	0.5	8 B
IRQ205	Channel 2 Interrupt	213	0xD5	205	0xCD	3688	0xE68	0.5	8 B
IRQ206	Channel 3 Interrupt	214	0xD6	206	0xCE	3696	0xE70	0.5	8 B
IRQ207	—	215	0xD7	207	0xCF	3704	0xE78	0.5	8 B
IRQ208	I^2C all	216	0xD8	208	0xD0	3712	0xE80	0.5	8 B
IRQ209	MAPLE BD 0	217	0xD9	209	0xD1	3720	0xE88	0.5	8 B
IRQ210	MAPLE BD 1	218	0xDA	210	0xD2	3728	0xE90	0.5	8 B
IRQ211	MAPLE BD 2	219	0xDB	211	0xD3	3736	0xE98	0.5	8 B
IRQ212	MAPLE BD 3	220	0xDC	212	0xD4	3744	0xEA0	0.5	8 B
IRQ213	MAPLE BD 4	221	0xDD	213	0xD5	3752	0xEA8	0.5	8 B
IRQ214	MAPLE BD 5	222	0xDE	214	0xD6	3760	0xEB0	0.5	8 B
IRQ215	MAPLE BD 6	223	0xDF	215	0xD7	3768	0xEB8	0.5	8 B
IRQ216	MAPLE BD 7	224	0xE0	216	0xD8	3776	0xEC0	0.5	8 B
IRQ217	MAPLE BD 8	225	0xE1	217	0xD9	3784	0xEC8	0.5	8 B
IRQ218	MAPLE BD 9	226	0xE2	218	0xDA	3792	0xED0	0.5	8 B
IRQ219	MAPLE BD 10	227	0xE3	219	0xDB	3800	0xED8	0.5	8 B
IRQ220	MAPLE BD 11	228	0xE4	220	0xDC	3808	0xEE0	0.5	8 B
IRQ221	MAPLE BD 12	229	0xE5	221	0xDD	3816	0xEE8	0.5	8 B
IRQ222	MAPLE BD 13	230	0xE6	222	0xDE	3824	0xEF0	0.5	8 B
IRQ223	MAPLE BD 14	231	0xE7	223	0xDF	3832	0xEF8	0.5	8 B
IRQ224	MAPLE BD 15	232	0xE8	224	0xE0	3840	0xF00	0.5	8 B
IRQ225	—	233	0xE9	225	0xE1	3848	0xF08	0.5	8 B
IRQ226	IRQ0	234	0xEA	226	0xE2	3856	0xF10	0.5	8 B
IRQ227	IRQ1	235	0xEB	227	0xE3	3864	0xF18	0.5	8 B
IRQ228	IRQ2	236	0xEC	228	0xE4	3872	0xF20	0.5	8 B

Table 13-5. Interrupt Summary Reference by Interrupt Indexes and VBA Offset (Continued)

Name	Description	Interrupt Index		EPIC Interrupt Index		Offset from VBA		Fetch Set	
		Dec	Hex	Dec	Hex	Dec	Hex	No.	Size
IRQ229	IRQ3	237	0xED	229	0xE5	3880	0xF28	0.5	8 B
IRQ230	IRQ4	238	0xEE	230	0xE6	3888	0xF30	0.5	8 B
IRQ231	IRQ5	239	0xEF	231	0xE7	3896	0xF38	0.5	8 B
IRQ232	IRQ6	240	0xF0	232	0xE8	3904	0xF40	0.5	8 B
IRQ233	IRQ7	241	0xF1	233	0xE9	3912	0xF48	0.5	8 B
IRQ234	IRQ8	242	0xF2	234	0xEA	3920	0xF50	0.5	8 B
IRQ235	IRQ9	243	0xF3	235	0xEB	3928	0xF58	0.5	8 B
IRQ236	IRQ10	244	0xF4	236	0xEC	3936	0xF60	0.5	8 B
IRQ237	IRQ11	245	0xF5	237	0xED	3944	0xF68	0.5	8 B
IRQ238	IRQ12	246	0xF6	238	0xEE	3952	0xF70	0.5	8 B
IRQ239	IRQ13	247	0xF7	239	0xEF	3960	0xF78	0.5	8 B
IRQ240	IRQ14	248	0xF8	240	0xF0	3968	0xF80	0.5	8 B
IRQ241	IRQ15	249	0xF9	241	0xF1	3976	0xF88	0.5	8 B
IRQ242	NMI	250	0xFA	242	0xF2	3984	0xF90	0.5	8 B
IRQ243	ORed TDM interrupts	251	0xFB	243	0xF3	3992	0xF98	0.5	8 B
IRQ244	ORed Debug Interrupts	252	0xFC	244	0xF4	4000	0xFA0	0.5	8 B
IRQ245	ORed General Interrupts	253	0xFD	245	0xF5	4008	0xFA8	0.5	8 B
IRQ246	ORed Watch Dog Timer Interrupts	254	0xFE	246	0xF6	4016	0xFB0	0.5	8 B
IRQ247	ORed MAPLE Interrupts	255	0xFF	247	0xF7	4024	0xFB8	0.5	8 B
IRQ248	Channel 0 Interrupt	256	0x100	248	0xF8	4032	0xFC0	0.5	8 B
IRQ249	Channel 1 Interrupt	257	0x101	249	0xF9	4040	0xFC8	0.5	8 B
IRQ250	Channel 2 Interrupt	258	0x102	250	0xFA	4048	0xFD0	0.5	8 B
IRQ251	Channel 3 Interrupt	259	0x103	251	0xFB	4056	0xFD8	0.5	8 B
IRQ252	—	260	0x104	252	0xFC	4064	0xFE0	0.5	8 B
IRQ253	—	261	0x105	253	0xFD	4072	0xFE8	0.5	8 B
IRQ254	—	262	0x106	254	0xFE	4080	0xFF0	0.5	8 B
IRQ255	—	263	0x107	255	0xFF	4088	0xFF8	0.5	8 B

13.4 Core Interrupt Mesh

To enhance communication between the six SC3850 DSP core subsystems, the MSC8156E has a core interrupt mesh. This mesh is built by connecting two interrupts from each of the SC3850 core subsystems to each of the DSP core subsystems (including itself). **Table 13-6** describes the interrupt names connecting the DSP core subsystems:

Table 13-6. Core Interrupt Mesh

From/To	Core Subsystem 0	Core Subsystem 1	Core Subsystem 2	Core Subsystem 3	Core Subsystem 4	Core Subsystem 5
Core Subsystem 0	M_GPR0[1:0]	M_GPR0[3:2]	M_GPR0[5:4]	M_GPR0[7:6]	M_GPR0[9:8]	M_GPR0[11:10]
Core Subsystem 1	M_GPR0[1:0]	M_GPR0[3:2]	M_GPR0[5:4]	M_GPR0[7:6]	M_GPR0[9:8]	M_GPR0[11:10]
Core Subsystem 2	M_GPR0[1:0]	M_GPR0[3:2]	M_GPR0[5:4]	M_GPR0[7:6]	M_GPR0[9:8]	M_GPR0[11:10]
Core Subsystem 3	M_GPR0[1:0]	M_GPR0[3:2]	M_GPR0[5:4]	M_GPR0[7:6]	M_GPR0[9:8]	M_GPR0[11:10]
Core Subsystem 4	M_GPR0[1:0]	M_GPR0[3:2]	M_GPR0[5:4]	M_GPR0[7:6]	M_GPR0[9:8]	M_GPR0[11:10]
Core Subsystem 5	M_GPR0[1:0]	M_GPR0[3:2]	M_GPR0[5:4]	M_GPR0[7:6]	M_GPR0[9:8]	M_GPR0[11:10]

Note: See the *SC3850 DSP Core Subsystem Reference Manual* for assertion/deassertion of the smmu_gpr signals. As an example of how to use this information, to assert an interrupt from core 1 to core 2, set bit 5 or 4 in M_GPR0 in the core 1 subsystem, which is the sending core.

13.5 Programming Model

The MSC8156E interrupt program model includes configuration of the global interrupt controller and the general configuration block interrupt registers.

Note: See the *SC3850 DSP Core Subsystem Reference Manual* for configuration and programming of the EPIC registers.

13.5.1 Global Interrupt Controller

The virtual interrupt registers reside in a 256-byte address space (for more information see **Chapter 9**, *Memory Map*) and include:

- Virtual Interrupt Generation Register (VIGR)
- Virtual Interrupt status register (VISR)

Note: The GIC registers use a base address of: 0xFFF27000.

13.5.1.1 Virtual Interrupt Generation Register (VIGR)

VIGR Virtual Interrupt Generation Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Type	W															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	VIRQNUM_H		—	—	—	—	—	VIRQNUM_L		
Type	W															

VIGR generates virtual interrupts according to the written data. The VIRQ generated corresponds to the combination of {VIRQNUM_H, VIRQNUM_L}. A read from VIGR returns all zeros. Notice that the supported values of {VIRQNUM_H, VIRQNUM_L} are 0 to 25.

Table 13-7. VIGR Bit Descriptions

Name	Description	Settings
— 31–10	Reserved. Write to zero for future compatibility.	
VIRQNUM_H 9–8	**Virtual Interrupt Number (High)** The high bits of the virtual interrupt index number.	00 to 11
— 7–3	Reserved. Write to zero for future compatibility.	
VIRQNUM_L 2–0	**Virtual Interrupt Number (Low)** The low bits of the virtual interrupt index number.	000 to 111

13.5.1.2 Virtual Interrupt Status Register (VISR)

VISR Virtual Interrupt Status Register ‘Offset 0x08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	—	—	—	—	—	VS25	VS24	VS23	VS22	VS21	VS20	VS19	VS18	VS17	VS16
Type	R/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
	VS15	VS14	VS13	VS12	VS11	VS10	VS9	VS8	VS7	VS6	VS5	VS4	VS3	VS2	VS1	VS0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the VISR corresponds to one virtual interrupt source, selected by proper write access to the VIGR. When the interrupt is generated by this write access, the GIC sets the corresponding status bit. It is the responsibility of the interrupt service routine (ISR) of the destination to clear only the correct status bits by writing ones to them. Writing zeros to status bits has no effect on their status. A set status bit does not block the generation of another virtual interrupt pulse by additional writes to VIGR.

Table 13-8. VISR Bit Descriptions

Name	Description	Settings
— 31–26	Reserved. Write to zero for future compatibility.	
VS25 25	**Virtual Interrupt 25 Status** Reflects the status of NMI_OUT.	0 Interrupt not asserted 1 Interrupt asserted
VS24 24	**Virtual Interrupt 24 Status** Reflects the status of INT_OUT.	0 Interrupt not asserted 1 Interrupt asserted
VS23 23	**Virtual Interrupt 23 Status** Reflects the status of VNMI_7.	0 Interrupt not asserted 1 Interrupt asserted
VS22 22	**Virtual Interrupt 22 Status** Reflects the status of VNMI_6.	0 Interrupt not asserted 1 Interrupt asserted
VS21 21	**Virtual Interrupt 21 Status** Reflects the status of tVNMI_5.	0 Interrupt not asserted 1 Interrupt asserted
VS20 20	**Virtual Interrupt 20 Status** Reflects the status of VNMI_4.	0 Interrupt not asserted 1 Interrupt asserted
VS19 19	**Virtual Interrupt 19 Status** Reflects the status of VNMI_3.	0 Interrupt not asserted 1 Interrupt asserted
VS18 18	**Virtual Interrupt 18 Status** Reflects the status of VNMI_2.	0 Interrupt not asserted 1 Interrupt asserted
VS17 17	**Virtual Interrupt 17 Status** Reflects the status of VNMI_1.	0 Interrupt not asserted 1 Interrupt asserted
VS16 16	**Virtual Interrupt 16 Status** Reflects the status of VNMI_0.	0 Interrupt not asserted 1 Interrupt asserted

Table 13-8. VISR Bit Descriptions (Continued)

Name	Description	Settings
VS15 15	**Virtual Interrupt 15 Status** Reflects the status of the core virtual interrupt 15.	0 Interrupt not asserted 1 Interrupt asserted
VS14 14	**Virtual Interrupt 14 Status** Reflects the status of the core virtual interrupt 14.	0 Interrupt not asserted 1 Interrupt asserted
VS13 13	**Virtual Interrupt 13 Status** Reflects the status of the core virtual interrupt 13.	0 Interrupt not asserted 1 Interrupt asserted
VS12 12	**Virtual Interrupt 12 Status** Reflects the status of the core virtual interrupt 12.	0 Interrupt not asserted 1 Interrupt asserted
VS11 11	**Virtual Interrupt 11 Status** Reflects the status of the core virtual interrupt 11.	0 Interrupt not asserted 1 Interrupt asserted
VS10 10	**Virtual Interrupt 10 Status** Reflects the status of the core virtual interrupt 10.	0 Interrupt not asserted 1 Interrupt asserted
VS9 9	**Virtual Interrupt 9 Status** Reflects the status of the core virtual interrupt 9.	0 Interrupt not asserted 1 Interrupt asserted
VS8 8	**Virtual Interrupt 8 Status** Reflects the status of the core virtual interrupt 8.	0 Interrupt not asserted 1 Interrupt asserted
VS7 7	**Virtual Interrupt 7 Status** Reflects the status of the core virtual interrupt 7.	0 Interrupt not asserted 1 Interrupt asserted
VS6 6	**Virtual Interrupt 6 Status** Reflects the status of the core virtual interrupt 6.	0 Interrupt not asserted 1 Interrupt asserted
VS5 5	**Virtual Interrupt 5 Status** Reflects the status of the core virtual interrupt 5.	0 Interrupt not asserted 1 Interrupt asserted
VS4 4	**Virtual Interrupt 4 Status** Reflects the status of the core virtual interrupt 4.	0 Interrupt not asserted 1 Interrupt asserted
VS3 3	**Virtual Interrupt 3 Status** Reflects the status of the core virtual interrupt 3.	0 Interrupt not asserted 1 Interrupt asserted
VS2 2	**Virtual Interrupt 2 Status** Reflects the status of the core virtual interrupt 2.	0 Interrupt not asserted 1 Interrupt asserted
VS1 1	**Virtual Interrupt 2 Status** Reflects the status of the core virtual interrupt 1.	0 Interrupt not asserted 1 Interrupt asserted
VS0 0	**Virtual Interrupt 0 Status** Reflects the status of the core virtual interrupt 0.	0 Interrupt not asserted 1 Interrupt asserted

13.5.2 General Interrupt Configuration

The general configuration block resides in a 128-byte address space (see **Chapter 9**, *Memory Map*). These registers are described in detail in **Chapter 8**, *General Configuration Registers*, and include the following interrupt configuration registers:

- General Interrupt Register 1 (GIR1), see **page 8-32**
- General Interrupt Enable Registers 1 for Cores 0–5 (GIER1_[0–5]), see **page 8-35**
- General Interrupt Register 3 (GIR3), see **page 8-37**
- General Interrupt Enable Registers 3 for Cores 0–5 (GIER3_[0–5]), see **page 8-39**
- General Interrupt Register 5 (GIR5), see **page 8-40**
- General Interrupt Enable Registers 5 for Cores 0–5 (GIER5_[0–5]), see **page 8-42**

Note: The general interrupt configuration registers use a base address of: 0xFFF28000.

13.5.3 Programming Restrictions

If a precise interrupt occurs in the SC3850 subsystem, the cause of the interrupt must be resolved before returning from the interrupt handler to normal code execution. If the interrupt is not resolved and cleared, an endless loop can occur and cause a deadlock. For details, see the *MSC8156 SC3850 DSP Subsystem Reference Manual*, **Appendix C**: *Error Handling*.

Direct Memory Access (DMA) Controller

14

The DMA controller enables data movement and rearrangement while the DSP cores work independently. It can transfer blocks of data to and from the M2 memory, M3 memory, and the DDR SDRAM controllers via the CLASS. It has 16 high-speed bidirectional channels and can be commanded from each of the DSP subsystems, as well as from up to two external initiators through the RapidIO or PCI Express interfaces using buffer descriptors (BDs). All channels are capable of complex data movement and advanced transaction chaining. Operations such as descriptor fetches and block transfers are initiated by each of the sixteen channels. Full duplex operation allows the DMA controller to read data from one target and store it in its internal memory while concurrently writing another buffer to another target. This capability can be used extensively when data is read from the M3 memory and written into the M2 memory. The bidirectional DMA controller reads from one of the CLASS target ports while writing to the second one. The DMA controller supports smart arbitration algorithms such as round-robin and a timer-based mechanism using an earliest deadline first (EDF) algorithm. The DMA controller also supports a Debug mode and profiling for application development and testing. **Figure 14-1** shows the VCOP block diagram.

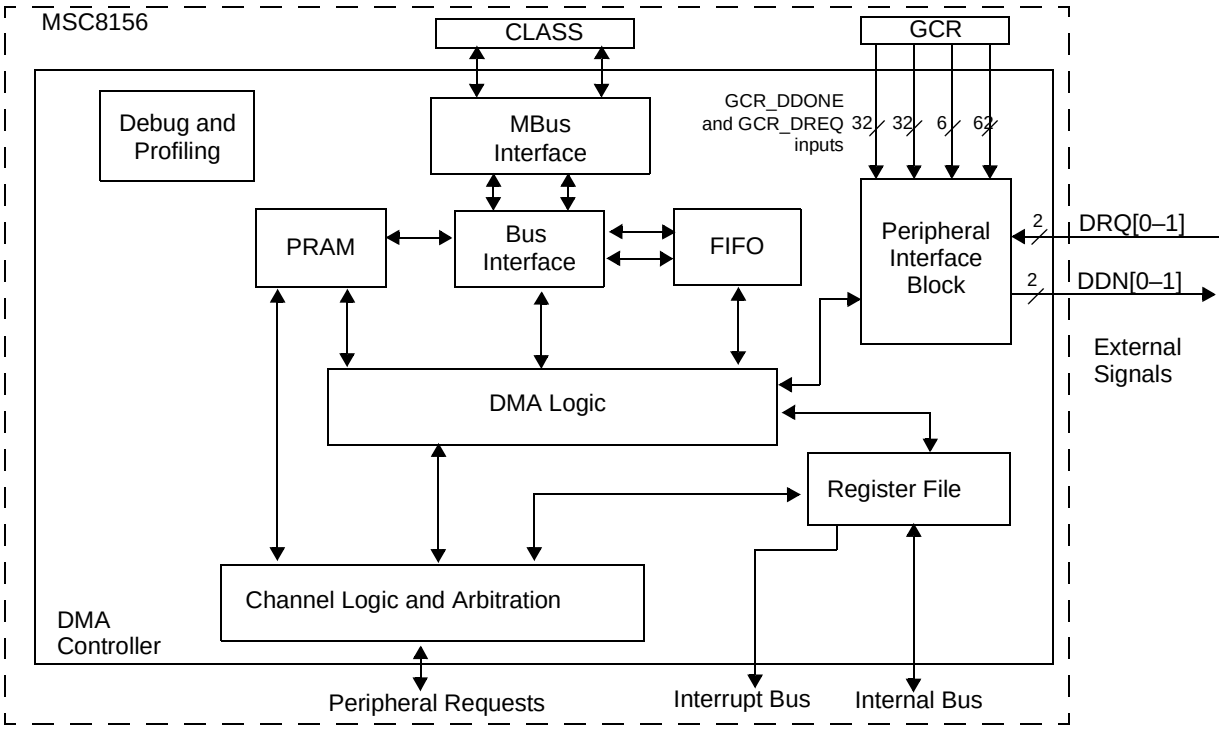


Figure 14-1. DMA Controller Block Diagram

14.1 Operating Modes

The DMA controller supports to modes of operation:

- *Functional mode*. Each of the data transactions can be executed on each of the MBus ports. The VCOP supports memory to memory data transfers.
- *Debug mode*. The VCOP enters the debug by external debug request. Once the VCOP enters the debug mode, it holds its requests to the MBus ports. The internal logic in the VCOP masks the channels.
  - MBus is in debug mode and each of its ports gracefully stops its transaction.
  - Channel logic in debug mode. The arbitration mechanism masks all channels requests. The last serviced channel gets is fully serviced.

14.2 Buffer Types

The DMA channel parameter RAM (PRAM) is accessible to the DMA controller and the port interface and includes a parity mechanism. Each channel has one dedicated PRAM line. The external BD is fetched into the PRAM the first time the channel wins during arbitration.

When a buffer is activated, the DMA controller generates a bus transaction with a maximum size as described in the buffer descriptor BTSZ field and decrements BD_SIZE accordingly. The

address can increment or freeze. When BD_SIZE reaches zero, the channel takes one of the following actions:

- Shuts down (simple buffer)
- Reinitializes itself (cyclic buffer)
- Reinstalls its size (incremental buffer)
- Switches to another buffer (chained-buffers)
- Any combination of the preceding
- Updates the multi-dimension parameters (multi dimension-buffers)

The sections that follow provide examples of several types of buffers. The BD_ATTR fields listed for each example are only those that do not have zero values.

14.2.1 One-Dimensional Simple Buffer

A simple channel is a buffer that closes when BD_SIZE reaches zero. It is defined by clearing the BD_ATTR[CONT] field to zero (see **Table 14-29, *BD_ATTR Field Descriptions,*** on page 14-49). **Figure 14-2** shows an example simple buffer.

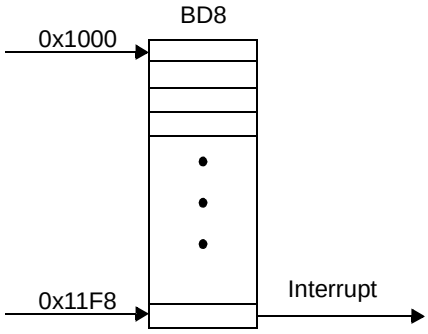


Figure 14-2. One-Dimensional Simple Buffer

Table 14-1 shows how a simple buffer, designated as BD8, is configured. A 0x200 byte block is read from address 0x1000. The channel closes when the data transfer is complete, and an interrupt is generated. Burst transactions are used on the bus.

Table 14-1. Channel Parameter Values for a Simple Buffer

BD	BD Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_SIZE		0x200	Size of transfer left for this buffer.
	BD_BSIZE		—	Buffer base size of cyclic buffer.
	BD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x0	Non-continuous mode: the buffer closes when the size reaches zero.
		CYC	0x0	Increment BD_ADDR when the size reaches zero.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.

14.2.2 One-Dimensional Cyclic Buffer

A cyclic buffer is a continuous buffer. When the buffer current address reaches zero, the pointer jumps back to the base address and the buffer executes again. **Figure 14-3** shows an example of a cyclic buffer.

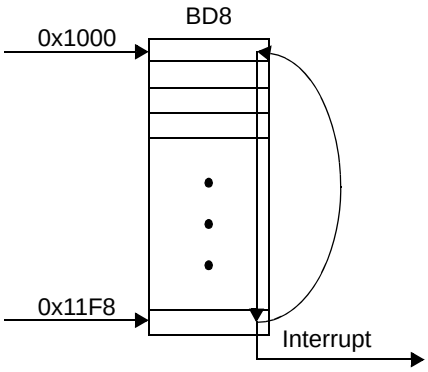


Figure 14-3. One-Dimensional Cyclic Buffer

Table 14-2 lists the channel parameters values for channel BD8 when a 0x200 byte block is read from address 0x1000. An interrupt is generated when the buffer size reaches zero, and the transfer restarts from the base address 0x1000.

Table 14-2. Channel Parameter Values for a Cyclic Buffer

BD	BD Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_SIZE		0x200	Size of transfer left for this buffer.
	BD_BSIZE		0x200	Buffer base size of cyclic buffer.
	BD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x1	Continuous mode: the buffer is not closed when the size reaches zero.
		CYC	0x1	Reinitialize BD_ADDR to original value when the size reaches zero.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.

14.2.3 One-Dimensional Chained Buffer

In a chained buffer scheme, when the size of the first buffer reaches zero, the read jumps to the address of the next buffer, which may be another chained buffer or another buffer type (simple, cyclic, or incremental)). **Figure 14-4** shows a buffer chained to a simple buffer.

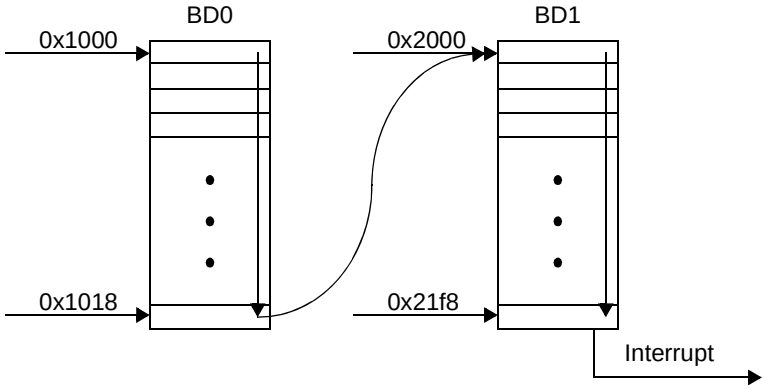


Figure 14-4. One-Dimensional Chained Buffer

There is no constraint on the port used by each chained buffer. However, if the buffers use different ports, the DMA controller masks requests until data is out of the source or in the destination. This operation prevents out-of-sequence transactions at the ports. **Table 14-3** lists the channel parameter values associated with a chained buffer (BD0) and a simple buffer (BD1). A 0x20 byte block is read starting from address 0x1000 (buffer 0). When the buffer size is zero, there is a jump to address 0x2000 (buffer 1). 0x200 byte blocks are read and an interrupt is generated.

Table 14-3. Channel Parameter Values for a Chained Buffer and a Simple Buffer

BD	BD Parameters		Value	Description
0	BD_ADDR		0x1000	External memory buffer current address.
	BD_SIZE		0x20	Size of transfer left for this buffer.
	BD_BSIZE		0x20	Buffer base size of cyclic buffer.
	BD_ATTR	CONT	0x1	Continuous mode. Do not close the buffer when size reaches zero.
		CYC	0x0	Non-cyclic.
		NBD	0x1	When size reaches zero, the next request calls buffer 1.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.
1	BD_ADDR		0x2000	External memory buffer current address.
	BD_SIZE		0x200	Size of transfer left for this buffer.
	BD_BSIZE		0x200	Buffer base size of cyclic buffer.
	BD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x0	Non-continuous mode. Close the buffer when size reaches zero.
		CYC	0x0	Non-cyclic mode.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.

14.2.4 One-Dimensional Incremental Buffer

In an incremental buffer, a data transfer starts at the buffer base address and continues until all data is transferred. An interrupt is generated each time BD_SIZE reaches zero. BD_ATTR[CONT] = 1, so the channel does not close when BD_SIZE reaches zero. BD_ATTR[CYC] = 0, signifying sequential addressing. NBD points to the buffer itself. **Figure 14-5** shows an example incremental buffer.

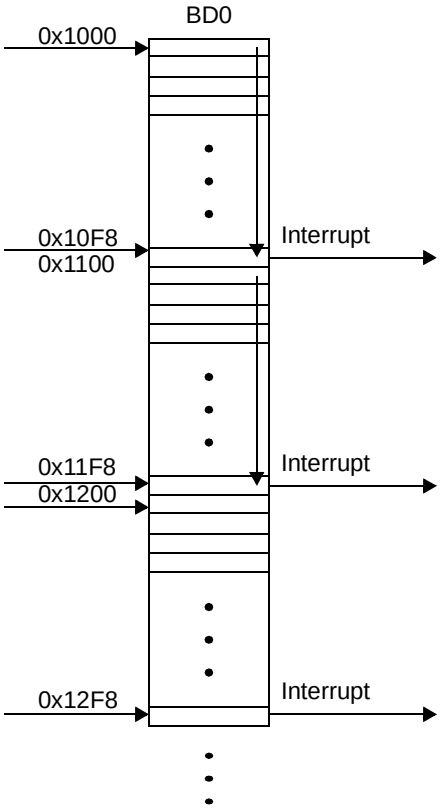


Figure 14-5. One-Dimensional Incremental Buffer

Table 14-4 lists the channel parameter values for an incremental buffer (BD0). Blocks of 0x100 bytes are read, starting at address 0x1000, and an interrupt is generated every 0x100 bytes. The mode is continuous and addressing is sequential. Be aware that in an incremental buffer, memory can be corrupted because of overwriting.

Table 14-4. Channel Parameter Values for an Incremental Buffer

BD	BD Parameters		Value	Description
0	BD_ADDR		0x1000	External memory buffer current address.
	BD_SIZE		0x100	Size of transfer left for this buffer.
	BD_BSIZE		0x100	Buffer base size of cyclic buffer.
	BD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x1	Continuous mode. Do not close the buffer when size reaches zero.
		CYC	0x0	Increment BD_ADDRESS when size reaches zero.
		NBD	0x0	Next request calls buffer 0 when size reaches zero.

14.2.5 One-Dimensional Complex Buffers With Dual Cyclic Buffers

Any combination of the previously described buffers can be used. Dual cyclic buffers, which use two areas in memory to store data, constitute a useful combination of buffer types. While one area of memory is processed, the other receives new data, as shown in **Figure 14-6**.

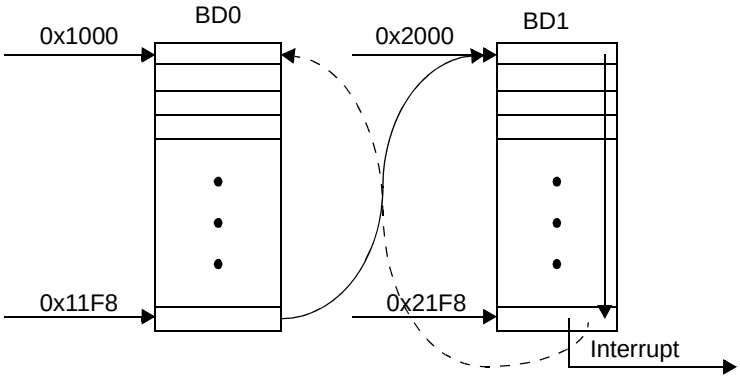


Figure 14-6. Dual Cyclic Buffers

Buffer 0 starts at address 0x1000, and transfers 0x200 byte-blocks. Buffer 1 starts at address 0x2000 and transfer size is also 0x200 bytes. **Table 14-5** lists the channel parameter values corresponding to dual cyclic buffers.

Table 14-5. Channel Parameter Values for Dual Cyclic Buffers

BD	BD Parameters		Value	Description
0	BD_ADDR		0x1000	External memory buffer current address.
	BD_SIZE		0x200	Size of transfer left for this buffer.
	BD_BSIZE		0x200	Buffer base size of cyclic buffer.
	BD_ATTR	CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero.
		CYC	0x1	Reinitialize BD_ADDRESS to original value when size reaches zero.
		NBD	0x1	When size reaches zero, next request calls buffer 1.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.
1	BD_ADDR		0x2000	External memory buffer current address.
	BD_SIZE		0x200	Size of transfer left for this buffer.
	BD_BSIZE		0x200	Buffer base size of cyclic buffer.
	BD_ATTR	SST	0x1	Generate interrupt when buffer ends
		CONT	0x1	Continuous mode. Do not shut down the channel when size reaches zero
		CYC	0x1	Reinitialize BD_ADDRESS to original value when size reaches zero
		NBD	0x0	When size reaches zero, the next request calls buffer 0
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes

14.2.6 Two-Dimensional Simple Buffer

A two-dimensional simple channel is a buffer that closes when BD_SIZE and M2D_COUNT reach zero. This buffer is defined as follows:

- BD_ATTR[CONT] = 0
- BD_MD_ATTR[BD] = 1
- DMACHCR[xMDC] = 1

M2D_COUNT must be set to the two-dimensional parameter and the M2D_OFFSET is set to the next address offset for each two-dimensional loop. The M2D_OFFSET is written in two's complement. The parameters of the third and fourth dimensions must be set to zero. **Figure 14-7** shows an example of a two-dimensional simple buffer.

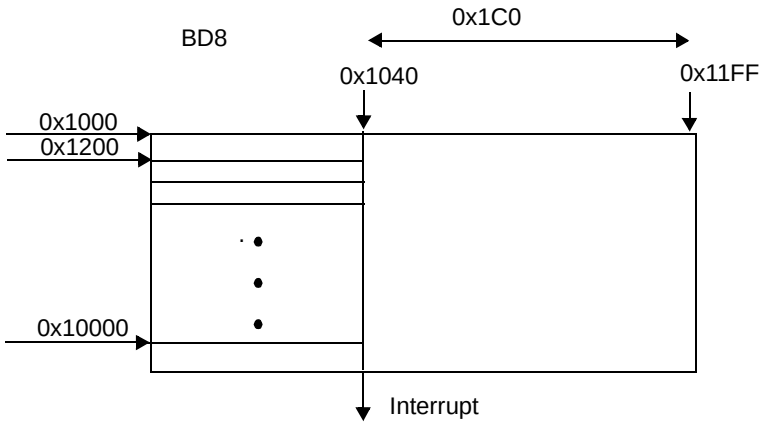


Figure 14-7. Two-Dimensional Simple Buffer

Table 14-6 lists the configuration of a simple buffer designated as channel BD8. A 0x2000 (0x80 × 0x40) byte two-dimensional block is read from address 0x1000. The first dimension is a line of 0x40 bytes. The second dimension is composed of 0x80 lines of 0x40 bytes each. The offset between each 0x40 byte transaction is 0x1c0. The channel closes when the transfer completes after 0x80 iterations, and an interrupt is generated. Burst transactions are used on the bus.

Table 14-6. Channel Parameter Values for a Two-Dimensional Simple Buffer

BD	BD Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x40	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x40	Buffer base size of continuous buffer.
	BD_MD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x0	Non-continuous mode: the channel closes when the size reaches zero.
		CYC	0x0	Increment BD_ADDR when the size reaches zero.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.
		BD	0x1	Buffer dimension is 2.
		SSTD	0x1	Interrupt issued at the end of the second dimension.
		CONTD	0x0	Simple buffer.
	BD_MD_2D	M2D_COUNT	0x80	Second dimension iterations left.
		M2D_BCOUNT	—	Second dimension base number of iterations.
		M2D_OFFSET	0x1C0	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0	Third dimension iterations left.
		M3D_BCOUNT	0	Third dimension base number of iterations.
		M3D_OFFSET	0	Third dimension offset between two consecutive iterations.
	BD_MD_4D	M4D_COUNT	0	Fourth dimension iterations left.
		M4D_OFFSET	0	Fourth dimension offset between two consecutive iterations.

14.2.7 Three-Dimensional Simple Buffer

A three-dimensional simple channel is a buffer that closes when BD_SIZE, M2D_COUNT, and M3D_COUNT reach zero. It is defined as follows:

- BD_ATTR[CONT] = 0
- BD_MD_ATTR[BD] = 2
- DMACHCR[xMDC] = 1

The M2D_COUNT and M3D_COUNT must be set to each dimension parameter. The M2D_OFFSET and M3D_OFFSET must be set to the next address offset for each dimension loop. The MxD_OFFSET is written in twos-complement form. The parameters of the fourth dimension must be cleared to zero. **Figure 14-8** shows a three-dimensional simple buffer.

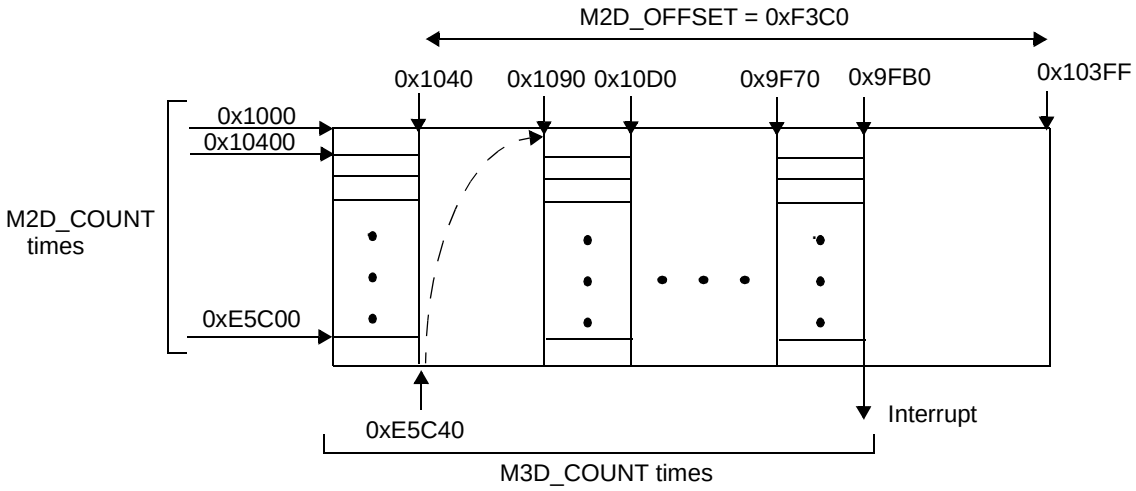


Figure 14-8. Three-Dimensional Simple Buffer

Table 14-7 shows the configuration of a simple buffer designated as channel BD8. A 0x40000 (0x100 × 10 × 40) three-dimensional block is read from address 0x1000. The basic buffer is 0x40 byte. The offset between each 0x40 byte transaction is 0xF3C0 (0x10400 – 0x1040). The second dimension parameter is 0x10. The offset between each two-dimensional buffers is –0xF4BB0 (0x1090 – 0xE5040). The channel closes when the transfer completes after 0x100 executions of the two-dimensional buffers, and an interrupt is generated. Burst transactions are used on the bus.

Table 14-7. Channel Parameter Values for a Three-Dimensional Simple Buffer

BD	BD Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x40	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x40	Buffer base size of continuous buffer.
	BD_MD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x0	Non-continuous mode: the channel is closed when the size reaches zero.
		CYC	0x0	Increment BD_ADDR when the size reaches zero.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes
		BD	0x2	Buffer dimension is 3.
		SSTD	0x2	Interrupt issued at the end of the third dimension.
		CONTD	0x0	Simple buffer.
	BD_MD_2D	M2D_COUNT	0x10	Second dimension iterations left.
		M2D_BCOUNT	0x10	Second dimension base number of iterations.
		M2D_OFFSET	0xF3C0	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0x100	Third dimension iterations left.
		M3D_BCOUNT	—	Third dimension base number of iterations.
		M3D_OFFSET	–0xE4BB0	Third dimension offset between two consecutive iterations of two-dimensional buffers.
	BD_MD_4D	M4D_COUNT	0	Fourth dimension iterations left.
		M4D_OFFSET	0	Fourth dimension offset between two consecutive iterations.

14.2.8 Four-Dimensional Simple Buffer

A four-dimensional simple channel is a buffer that closes when BD_SIZE and all MxD_COUNT reach zero. It is defined as follows:

- BD_ATTR[CONT] = 0
- BD_MD_ATTR[BD] = 3
- DMACHCR[xMDC] = 1

All MxD_COUNT must be set to their corresponding dimension parameter. All MxD_OFFSET must be set to the next address offset for the corresponding dimension loop. The MxD_OFFSET is written in twos-complement form. **Figure 14-9** shows an example four-dimensional simple buffer.

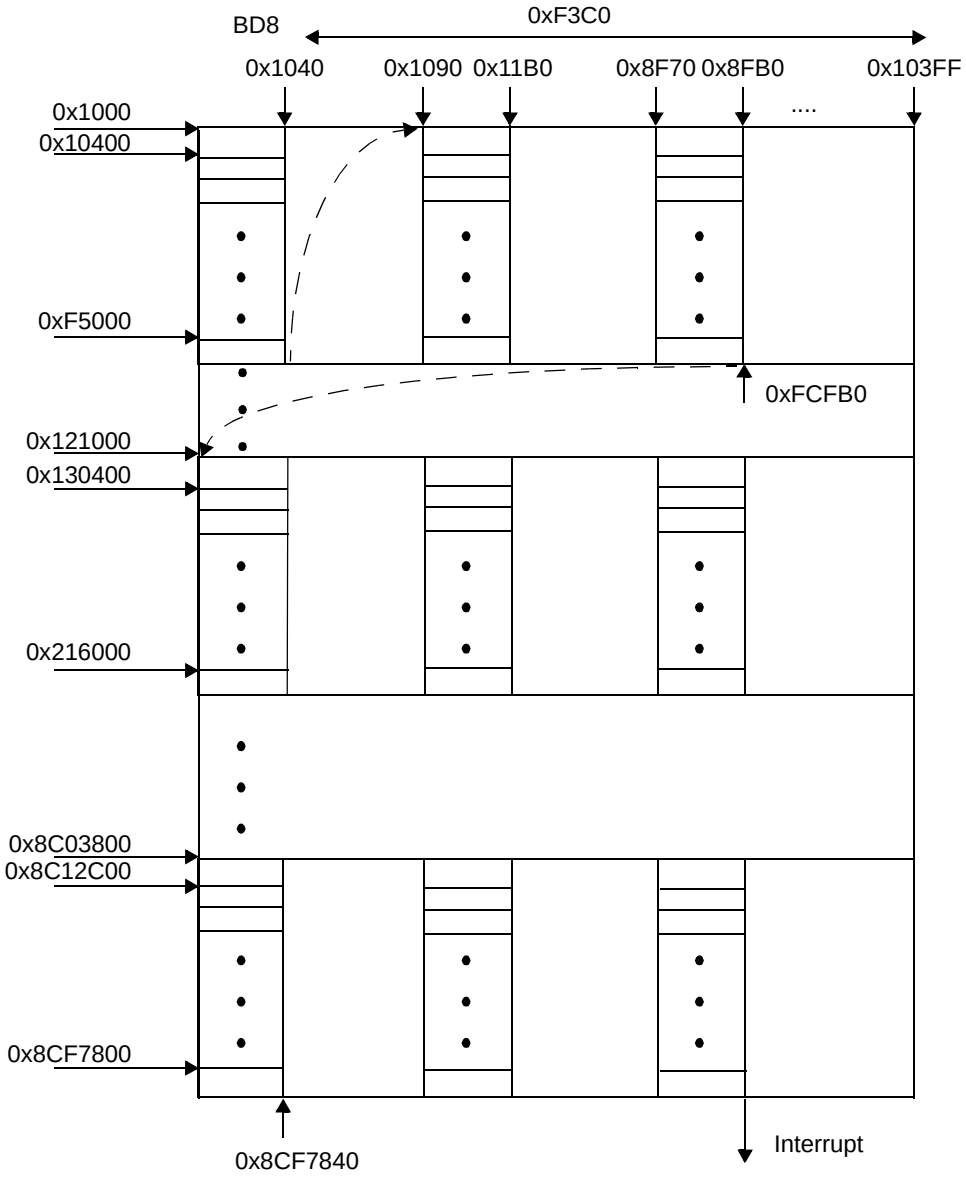


Figure 14-9. Four-Dimensional Simple Buffer

Table 14-8 lists the configuration of a simple buffer designated as channel BD8. A 0x2000000 (0x80 × 0x100 × 0x10 × 0x40) byte block is read from address 0x1000. The first dimension is a 0x40 byte buffer. The offset between each 0x40 bytes transaction is 0xF3C0. The two-dimensional buffers execute 0x100 times for each fourth dimension iteration. The offset between each two-dimensional buffers is –0xF3FB0 (0x1090 – 0xF5040). The channel closes when the transfer completes after 0x80 iterations of the three-dimensional buffer, and an interrupt is generated. Burst transactions are used on the bus.

Table 14-8. Channel Parameter Values for a Four Dimensional Simple Buffer

BD	BD Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x40	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x40	Buffer base size of continuous buffer.
	BD_MD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x0	Non-continuous mode: the channel closes when the size reaches zero.
		CYC	0x0	Increment BD_ADDR when the size reaches zero.
		BTSZ	0x7	Maximum transfer size is one burst of 64 bytes.
		BD	0x3	Buffer dimension is 4.
		SSTD	0x3	Interrupt issued at the end of the fourth dimension.
		CONTD	0x0	Simple buffer.
	BD_MD_2D	M2D_COUNT	0x10	Second dimension iterations left.
		M2D_BCOUNT	0x10	Second dimension base number of iterations.
		M2D_OFFSET	0xF3C0	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0x100	Third dimension iterations left.
		M3D_BCOUNT	0x100	Third dimension base number of iterations.
		M3D_OFFSET	0xF3FB0	Third dimension offset between two consecutive iterations.
	BD_MD_4D	M4D_COUNT	0x80	Fourth dimension iterations left.
		M4D_OFFSET	0x24050	Fourth dimension offset between two consecutive iterations.

14.2.9 Multi-Dimensional Chained Buffer

A multi-dimensional chained buffer has two or more multi-dimensional buffers. When the size of the first buffer and all its dimension counters reaches zero, the read jumps to the address of the next buffer, which may be another multi-chained buffer or another type of multi-dimensional buffer types (simple, cyclic, or incremental). The chained multi-dimensional buffers can be of any dimension. **Figure 14-10** shows a three-dimensional buffer chained to a four-dimensional simple buffer.

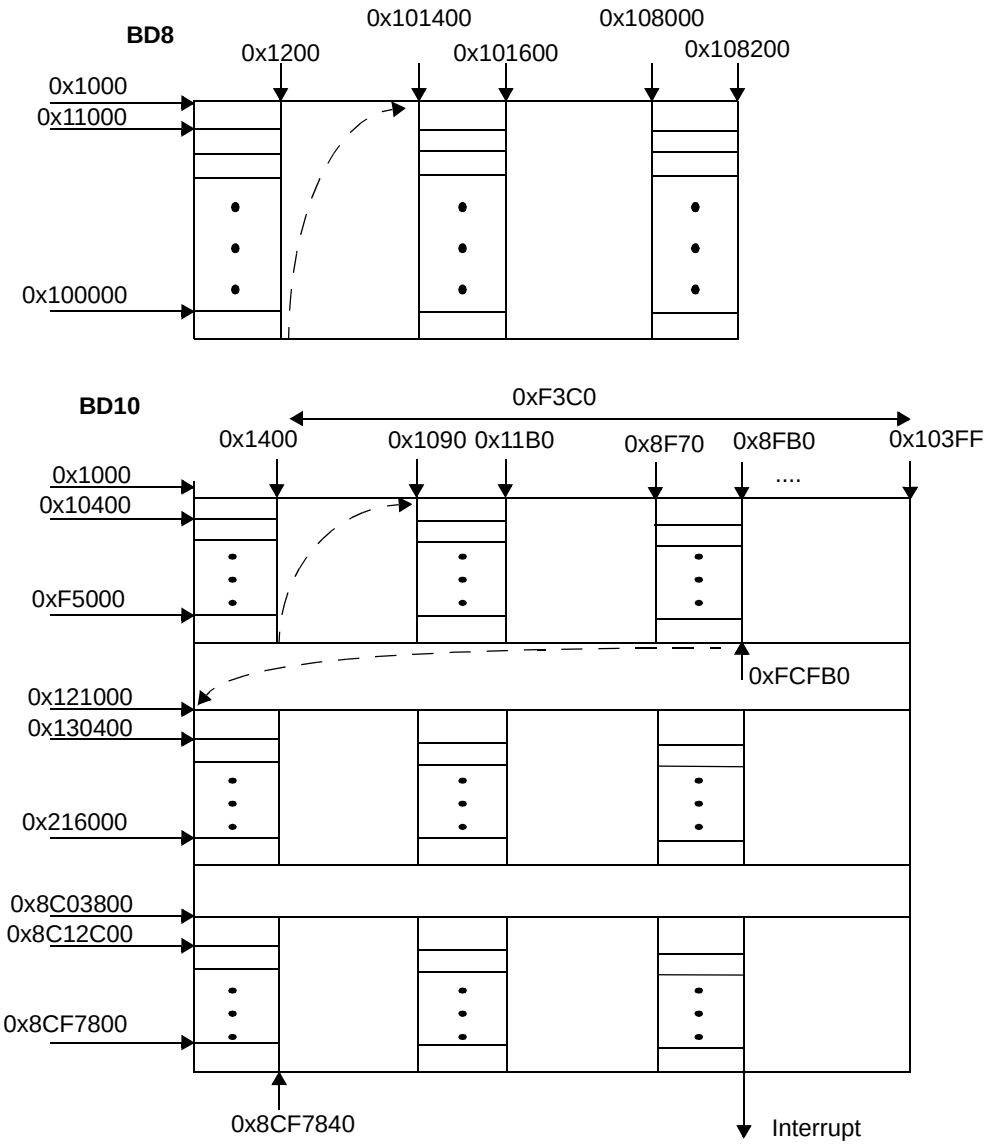


Figure 14-10. Multi-Dimensional Chained Buffer

There is no constraint on the port used by each chained buffer. However, if the buffers use different ports, the DMA logic masks requests until data is out of the source or in the destination. This operation prevents out-of-sequence transactions at the ports. **Table 14-9** shows the channel

parameter values associated with a three-dimensional chained buffer (BD8) and a four-dimensional simple buffer (BD10).

Table 14-9. Parameter Values for Multi-Dimensional Chained and Simple Buffers

BD	DCPRAM Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x200	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x200	Buffer base size of continuous buffer.
	BD_MD_ATTR	BTSZ	0x5	Basic transfer size is 16 bytes.
		NBD	0xA	Next Buffer descriptor is 10.
		SST	0x0	Do not generate interrupt when buffer ends.
		CONT	0x1	Continuous mode: the channel continues when the size and third dimension counter reach zero. See the CONTD field of the BD_MD_ATTR.
		CYC	0x0	Increment BD_ADDR when the size reaches zero.
		BD	0x2	Buffer dimension is 3.
		SSTD	0x0	No interrupt issued.
		CONTD	0x2	Continuous buffer when size reaches zero at the end of the third dimension.
	BD_MD_2D	M2D_COUNT	0x10	Second dimension iterations left.
		M2D_BCOUNT	0x10	Second dimension base number of iterations.
		M2D_OFFSET	0xFE00	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0x8	Third dimension iterations left.
		M3D_BCOUNT	-	Third dimension base number of iterations.
		M3D_OFFSET	0x1200	Third dimension offset between two consecutive iterations.
	BD_MD_4D	M4D_COUNT	0	Fourth dimension iterations left.
		M4D_OFFSET	0	Fourth dimension offset between two consecutive iterations.
10	BD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x40	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x40	Buffer base size of cyclic buffer.
	BD_ATTR	BTSZ	0x5	Basic transfer size is 16 bytes.
		SST	0x1	Generate interrupt when buffer ends.
		CONT	0x0	Non-continuous mode: the channel closes when the size reaches zero.
		CYC	0x0	Increment BD_ADDR when the size reaches zero.
	BD_MD_ATTR	BD	0x3	Buffer dimension is 4.
		SSTD	0x3	Interrupt issued at the end of the fourth dimension.
		CONTD	0x0	Simple buffer.
	BD_MD_2D	M2D_COUNT	0x10	Second dimension iterations left.
		M2D_BCOUNT	0x10	Second dimension base number of iterations.
		M2D_OFFSET	0xF3C0	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0x100	Third dimension iterations left.
		M3D_BCOUNT	0x100	Third dimension base number of iterations.
		M3D_OFFSET	–0xF3FB0	Third dimension offset between two consecutive iterations.
	BD_MD_4D	M4D_COUNT	0x80	Fourth dimension iterations left.
		M4D_OFFSET	0x24050	Fourth dimension offset between two consecutive iterations.

14.2.10 Two-Dimensional Cyclic Buffer

A two-dimensional cyclic buffer is a two-dimensional continuous buffer. When the size of the current buffer reaches zero, the pointer jumps back to the base address and the buffer executes again. The third and fourth dimension counters must be cleared to zero. The M3D_OFFSET must be set to the offset between the base address and the last transaction address. **Figure 14-11** shows an example cyclic buffer.

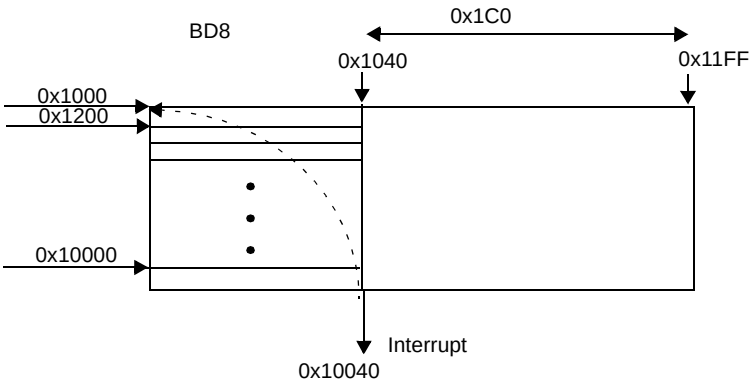


Figure 14-11. Two-Dimensional Cyclic Buffer

Table 14-10 lists the configuration of a two-dimensional cyclic buffer, designated as channel BD8 in this example. A 0x2000 (0x80 × 0x40) byte two dimension block is read from address 0x1000. The first dimension is a line of 0x40 bytes. The second dimension is a 0x80 lines of 0x40 bytes each. The offset between each 0x40 bytes transaction is 0x1C0. The base address is restored when the transfer is complete after 0x80 iterations.

Table 14-10. Channel Parameters Values for a Two Dimensions Cyclic Buffer

BD	BD Parameters		Value	Description
8	BD_MD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x40	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x40	Buffer base size of continuous buffer.
	BD_MD_ATTR	SST	0x0	Do not generate interrupt when buffer ends.
		CYC	0x1	Cyclic two-dimensional buffer; the third dimension offset is used to restore the base address.
		CONT	0x1	Continuous mode. The buffer does not close when BD_MD_BSIZE and M2D_COUNT reach zero.
		BTSZ	0x5	Basic transfer size is 16 bytes.
		BD	0x1	Buffer dimension is 2.
		CONTD	0x1	Second dimension continuous.
	BD_MD_2D	M2D_COUNT	0x80	Second dimension iterations left.
		M2D_BCOUNT	0x80	Second dimension base number of iterations.
		M2D_OFFSET	0x1C0	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0	Third dimension iterations left.
		M3D_BCOUNT	0	Third dimension base number of iterations.
		M3D_OFFSET	–0xF040	Third dimension offset between two consecutive iterations.
	BD_MD_4D	M4D_COUNT	0	Fourth dimension iterations left.
		M4D_OFFSET	0	Fourth dimension offset between two consecutive iterations.

14.2.11 Three-Dimensional Cyclic Buffer

A three-dimensional cyclic channel is a buffer that continues when BD_SIZE, M2D_COUNT, and M3D_COUNT reach zero. It is defined as follows:

- BD_ATTR[CONT] = 0
- BD_MD_ATTR[BD] = 2
- DMACHCR[xMDC] = 1

M2D_COUNT and M3D_COUNT must be set to each dimension parameter. The M2D_OFFSET and M3D_OFFSET must be set to the next address offset for each dimension loop. The M4D_OFFSET must be set to restore the base address of the channel. The MxD_OFFSET is written in two's complement. The counters of the fourth dimensions must be cleared to zero. **Figure 14-12** shows an example three-dimensional cyclic buffer.

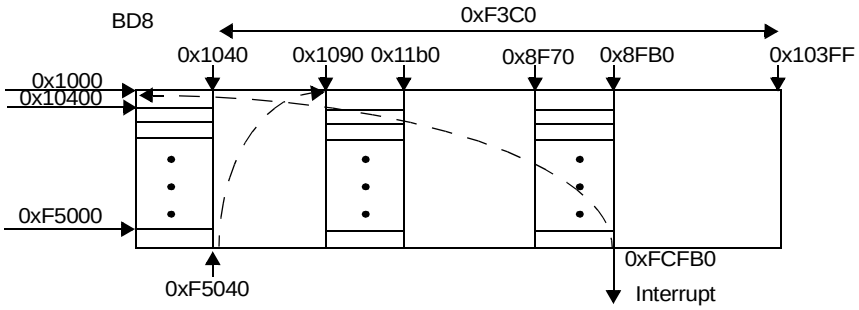


Figure 14-12. Three-Dimensional Cyclic Buffer

Table 14-11 shows the configuration of a cyclic buffer designated as channel BD8. A 0x40000 ($0x100 \times 10 \times 40$) three-dimensional block is read from address 0x1000. The basic buffer is 0x40 byte. The offset between each 0x40 byte transaction is 0xF3C0 (0x10400 – 0x1040). The two-dimensional parameter is 0x10. The offset between each two-dimensional buffer is –0xF3FB0 (0x1090 – 0xF5040). The base address of the channel is restored when the transfer completes after 0x100 executions of the two-dimensional buffers, and an interrupt is generated.

Table 14-11. Parameter Values for a Three-Dimensional Cyclic Buffer

BD	BD Parameters		Value	Description
8	BD_ADDR		0x1000	External memory buffer current address.
	BD_MD_SIZE		0x40	Size of transfer left for this buffer.
	BD_MD_BSIZE		0x40	Buffer base size of continuous buffer.
	BD_MD_ATTR	SST	0x1	Generate interrupt when buffer ends.
		CONT	0x1	Non-continuous mode: the channel closes when the size reaches zero.
		CYC	0x1	Cyclic three dimensions.
		BTSZ	0x5	Basic transfer size is 16 bytes.
		BD	0x2	Buffer dimension is 3.
		SSTD	0x2	Interrupt issued at the end of the third dimension.
		CONTD	0x2	Cyclic three-dimensional buffer.
	BD_MD_2D	M2D_COUNT	0x10	Second dimension iterations left.
		M2D_BCOUNT	0x10	Second dimension base number of iterations.
		M2D_OFFSET	0xF3C0	Second dimension offset between two consecutive iterations.
	BD_MD_3D	M3D_COUNT	0x100	Third dimension iterations left.
		M3D_BCOUNT	0x100	Third dimension base number of iterations.
		M3D_OFFSET	–0xF3FB0	Third dimension offset between two consecutive iterations of two dimension buffers.
	BD_MD_4D	M4D_COUNT	0	Fourth dimension iterations left.
		M4D_OFFSET	–0xFBFB0	Fourth dimension offset between two consecutive iterations.

14.3 Arbitration Types

There are two types of DMA arbitration: round-robin and early-deadline-serve-first (EDF). The type of arbitration is selected via the DGCR[AT] bit (see **Table 14-17, *DMAGCR Field Descriptions,*** on page 14-31).

14.3.1 Round-Robin Arbitration

The round-robin arbitration between channels is a least recently used (LRU) schema. Following is a list of the arbitration parameters according to their weight:

- *DMA port*. Each channel is assigned to one DMA port. Each time a channel request is serviced, the channels assigned to its port are masked for three clock cycles. When there are requesting channels for both ports, they are serviced intermittently.
- *Fixed-priority among round-robin groups*. Each channel is assigned to one of the four priority groups as defined by the DCHCRx[RRPG] bit. Each priority group can contain from 0 (empty) to all 16 channels. Pending requests from the highest-priority group are serviced first.

- *Bandwidth control*. Each channel has credit for maximum consecutive grants according to BD_ATTR [TSZ] and BD_ATTR[BTSZ]. If TSZ is greater than BTSZ bytes, the channel wins consecutively in BTSZ byte portions until TSZ is reached. The channel may stop requesting before it is continuously granted the maximum transfer size. The channel keeps its priority until the address is aligned. The channel priority is updated when the buffer or dimension ends.
- *Least recently used round-robin*. A linear queue defines the channel priority. After reset, each channel has a unique priority according to its number. After a channel is serviced, it gets the lowest priority. All the other channels with a priority lower than the winning priority upgrade their priority. All the channels with higher priority on this cycle keep their priority so that they are guaranteed service. Some channels may get the same priority by the time. **Table 14-12** lists the priority of the channels for two successive cycles.

Table 14-12. Round-Robin Arbitration Example

Channel Number	Channel Request	Clock n Priority		Clock n+1 Priority
1	Deasserted	0 (Highest)		0 (Highest)
8	Deasserted	1		1
3	Asserted	2	Channel 3 win	31 (Lowest)
2	Asserted	3		2
6	Asserted	4		3
5	Deasserted	5		4
4	Deasserted	6		5
7	Deasserted	7 (Lowest)		6

14.3.2 EDF Arbitration

EDF arbitration optimizes the DMA transactions in a time domain, simplifying application development. The EDF algorithm assumes that the application needs certain data to be transferred within a certain time. Every channel declares its deadline target, and the DMA controller sorts all channels into four priority groups. The deadline is the time between the current counter value (DMAEDFTDLx[CC]) to the threshold value (DMAEDFTDLx[TH]). See page 14-34 for details. The features of EDF arbitration are as follows:

- Round-robin arbitration with channels in the same group.
- 8-bit counter and base register for each channel.
- Counter is enabled/disabled when channel is activated/deactivated.
- Two options for continued buffer:
  - *Continuous mode*. Continues the deadline counter and channel with no action by the EDF logic.
  - *Reset mode*. Reloads the counter.

- Maskable interrupt for threshold deadline crossing when an active counter crosses the threshold.
- Four optional clock sources for the counters and a DMA predivider.
- Automatic channel priority group supporting DMA based on EDF algorithm.

The EDF sorts the channels into four priority groups according to their time to deadline value. The arbitration between the groups is fixed-priority (lowest group number has the highest priority). The arbitration among the channels in the same group is round-robin. Pairs of channels with same priority in the same priority group are served according to their channel number, as illustrated in **Table 14-13** and **Table 14-14**.

Table 14-13. Channels Sorted Into Four Priority Groups

Time to Deadline	Priority Group
0–1	0
2–7	1
8–63	2
64–255	3

Table 14-14. Example of Channel Priority Sorting

Channel	Current Count	Threshold	Time to Deadline	Priority Group	Priority (n)	Priority (n+1)	Priority (n+2)	Priority (n+3)
0	100	0	100	3	1	0 (winner)	31	30
1	255	80	175	3	2	1	0 (winner)	31
2	255	80	175	3	2	1	0	0 (winner)
3	240	160	80	3	0 (winner)	31	30	29

The first priority parameter is the port, which changes each cycle. The second parameter is the time to deadline, which determines the channel group. The last parameter is the channel number, which gives higher priority to the lower channel number for channels in the same group with the same priority. After each channel is serviced, all priorities are updated on a round-robin basis.

14.3.2.1 Issuing Interrupts

The EDF logic can issue a maskable interrupt request for each counter. The EDF issues its interrupts on the error request line of the DMA controller. There is one source for EDF interrupts: the threshold violation for each counter, as specified in the DMA EDF Status Register (DMAEDFSTR) (see page 14-38). The EDF logic compares each counter value with the threshold value. If a counter value equals the threshold value, EDF logic sets the corresponding sticky bit in the pending register.

14.3.2.2 Counter Control

The EDF field in the source BD_ATTR of the channel defines the EDF logic behavior when source BD_SIZE reached zero.

14.3.2.3 Clock Source to the Counters

All the counters share the same clock source. There are four clock sources: 3 external (to the DMA) clock sources and the DMA clock divided by 16.

14.4 Interrupts

The DMA controller uses two types of interrupts: maskable and nonmaskable interrupts.

14.4.1 Maskable Interrupts

The DMA controller can issue one maskable interrupt per channel at the end of a buffer or end of a transfer. For multi-dimension buffers, the interrupt may be issued on any of the dimensions. For each maskable interrupt request, a bit in the DMA Status Register (DMASTR) indicates the interrupt source. The interrupt mask is configured via bits in DMA Mask Register (DMAMR). Maskable interrupts are output as 16 individual interrupts, one for each unidirectional destination channel.

Most of the maskable interrupts are issued to request service or indicate that a transfer is available. The exception is EDF threshold violation error interrupt. This interrupt can be masked for each counter.

14.4.2 Nonmaskable Interrupts

Except for the DMA counters threshold violation, nonmaskable interrupts are error interrupts and are all output by one level-error interrupt. The DMA controller issues unmasked interrupts for one of the following sources:

- EDF violation (the only maskable source of a DMA error; it can be masked per counter)
- Port 0 transfer error
- Port 1 transfer error
- Any parity error
- Buffer size of zero

Note: In some cases in which a PRAM parity error occurs, the BD size zero error may also be set.

For each error source, a bit in the DMA Error Register (DMAERR) indicates the error source. DMAERR also samples the first channel that caused the first bus error and the first channel that caused the first parity error.

14.5 Profiling

The DMA supports system-level profiling via the Channel Profiled (CHAPRO) and Destination Channel Profiled (DEST) bits of the DMA Local Profiling Configuration Register (DMALPCR) (see **Table 14-26, *DMALPCR Field Descriptions,*** on page 14-43). These bits provide the following indications:

- DMA channel active.
- Arbitration winner.
- End of buffer for a DMA channel.
- Bus request.
- Consecutive grant.

14.6 DMA Peripheral Interface

The DMA peripheral interface supports the handshaking signals that permit up to 2 devices to control the DMA transfers through the RapidIO or PCI Express interfaces using buffer descriptors (BDs). For each interface, the peripheral can generate a request using the DRQ0 or DRQ1 input signal. The DMA controller generates the appropriate DDN0 or DDN1 response to the peripheral device when the transfer is completed. Channel definition is configured via the registers in the GCR block (see **Chapter 8**, *General Configuration Registers* for details). Each peripheral can connect to any DMA channel. Internally, the peripheral requests are translated to MBus transactions and the peripheral has an individual request number generated when the DMA generates the transaction on the MBus. The simple asynchronous interface supports all types of buffers and multi-dimensional channels.

14.6.1 Modes of Operation

The DMA peripheral interface supports the following combinations of data transfers:

- *Peripheral to memory*. Source transactions are controlled by the data request (DRQn). After the channel is enabled, the destination BD is fetched. Read data arbitration is sustained until DRQn is asserted. As long as the signal remains asserted, data is read from the source to the DMA internal FIFO. Destination transactions depend only on the data in the DMA internal FIFO and the destination and channel programming. If DRQn is deasserted before the end of the BD, any previously won read transaction arbitrations are executed. Write transactions are executed as long as there is a minimum of 64 bytes of data in the DMA internal FIFO or the FIFO is being flushed due to BD_ATTR[MR] or port switching. Once the channel is closed or BD_ATTR[SST] is set, the done signal is asserted.
- *Memory to peripheral*. Destination transactions are controlled by the data request (DRQn). After the channel is enabled, the source and destination BDs are fetched. Write data

arbitration is sustained until the done signal is asserted. As long as the signal remains asserted, data is written from the DMA internal FIFO to the destination. Destination transactions depend on the data in the DMA internal FIFO, the state of the done signal, and the destination BD. Source data is arbitrated depending on the source BD, channel programming, and the internal DMA FIFO space. If DRQn is deasserted before the end of the BD, any previously won write transaction arbitrations are executed. Write transactions are arbitrated as long as there is a minimum of 64 bytes of data in the DMA internal FIFO or the FIFO is being flushed due to BD_ATTR[MR] or port switching and the done signal is asserted. Once the channel is closed or BD_ATTR[SST] is set and the BD is finished, the done signal is asserted.

- *Peripheral to peripheral*. Both the source and destination transactions are controlled by DRQn. They can be controlled by the same done signal or by different done signals.
  - Same done signal. Do not use this mode. If the associated channel source generates a done signal, then data will be left in FIFO. If the associated channel destination generates the done signal, source data will be fetched after the done is asserted until you flush the FIFO.
  - Different done signals. One data request signal is associated with the destination and a different request signal is associated with the source of the channel. This means that after the channel is enabled, the source and destination BDs are fetched. The read data arbitration operates in the same way as for the case of peripheral to memory. The write data arbitration behave in the same way as for the case of memory to peripheral. If the source data request is deasserted before the end of the BD, then previously won read transaction arbitrations are executed. If the destination request is deasserted before the end of the BD, any previously won write transaction arbitrations are executed. However, data may be left in the FIFO because no more write arbitrations can occur until the destination request is asserted again.

14.6.2 Configuration and Control Registers

The operation of DMA peripheral interface is configured using three of the General Configuration Registers:

- GCR DMA Request 0 (GCR_DREQ0)
- GCR DMA Request 1 (GCR_DREQ1)
- GCR DMA Done (GCR_DDONE)

For details about the layout and structure of these registers, see **Chapter 8**, *General Configuration Registers*.

14.6.3 Functional Description

The DMA peripheral interface block controls the operation of the request and done signals for two peripherals. The following subsections describe the how the signals are handled.

14.6.3.1 Request Signal

The request signals (initiated by the peripheral by asserting the appropriate DRQn input) are synchronized by an internal clock and control the internal associated DMA channel requests. For a source channel, once the DMA channel is enabled and the associated request signal is asserted, the channel reads data from the source. For a destination channel, once the DMA channel is enabled and the associated request signal is asserted, the channel writes data to the destination. The input signal does not enable the channel or the channel logic pulse that sets the channel bit in the DMASTR. However, deasserting the signal disables future arbitration wins for the associated channel. Previously won arbitrations continue to the end even after the request signal is pulled low.

14.6.3.2 Done Signal

The done signals are driven by an internal channel logic pulse width that sets the corresponding channel bit in the DMASTR. You must always set DMA_BD_ATTR[SST] and DMA_BD_ATTR[MR] to enable the generation of the done signal at the end of the BD (or the selected dimension for multi-dimensional channels). When enabled, the interface generates a done signal (which is expressed on the appropriate DDNn output line).

14.6.3.3 Signal Operation

The request and done signal operate under the following conditions:

- The done signal is generated when the DMA execution reaches the end of the BD or the channel is closed.
- The done signal is driven by one channel only.
- Once the done signal is asserted, it is not deasserted as long as the corresponding request signal is asserted.
- After the done signal is asserted, additional requests for the channel are ignored until the request signal is deasserted, the done signal is deasserted, and the request signal is reasserted.

14.6.4 Using the DMA Peripheral Interface Block

To use the DMA peripheral interface block features, you must configure and initialize the block using the following procedures:

1. Program the DMA_BD and other DMA registers for memory-to-memory transactions.
2. Set the corresponding DMA_BD_ATTR[SST] and DMA_BD_ATTR[MR} and, for multidimensional operation BD_MD_ATTR[SSTD] and BD_MD_ATTR[MRD], as required by the peripheral.
3. Because they are multiplexed with GPIO signals, you must configure the GPIO interface to specify the DDN[0–1] and DRQ[0–1] signals. The following list identifies the appropriate GPIO signals to configure:
   - GPIO3 must be configured as DRQ1
   - GPIO4 must be configured as DDN1
   - GPIO14 must be configured as DRQ0
   - GPIO15 must be configured as DDN0

   See **Chapter 24**, *GPIO* for details.
4. You must configure the channels to associate with the signals using GCR_DREQ0, GCR_DREQ1, and GCR_DDONE. See **Chapter 8**, *General Configuration Registers* for details.
5. Enable the channel.

After the MSC8156E has set up the DMA controller, the peripheral must perform the following steps:

1. Deassert the DRQn input signal until the peripheral is ready to initiate the transfer.
2. When ready, assert DRQn.
3. Deassert DRQn when DDNn asserts either because it is ready for the next BD or dimension or it has finished the transaction.
4. Reassert DRQn if there is more data to transmit or space to receive the next BD.
5. Continue on in this manner until all transactions are complete.

14.7 DMA Programming Model

The DMA controller uses a combination of registers used to configure and report status of the DMA transfers and the Buffer Descriptor tables that control and implement the DMA transfers.

- DMA Registers
  - DMA Buffer Descriptor Base Registers 0–15 (DMABDBR[0–15], page 14-28
  - DMA Controller Channel Configuration Registers 0–15 (DMACHCR[0–15]), page 14-29
  - DMA Controller Global Configuration Register (DMAGCR), page 14-31
  - DMA Channel Enable Register (DMACHER), page 14-31
  - DMA Channel Disable Register (DMACHDR), page 14-32
  - DMA Channel Freeze Register (DMACHFR),
  - DMA Channel Defrost Register (DMACHDFR),
  - DMA EDF Time-to-Dead Line Registers 0–15 (DMAEDFDL[0–15]), page 14-34
  - DMA EDF Control Register (DMAEDFCTRL), page 14-35
  - DMA EDF Mask Register (DMAEDFMR), page 14-35
  - DMA EDF Mask Update Register (DMAEDFMUR), page 14-36
  - DMA EDF Status Register (DMAEDFSTR), page 14-38
  - DMA Mask Register (DMAMR), page 14-38
  - DMA Mask Update Register (DMAMUR), page 14-39
  - DMA Status Register (DMASTR), page 14-40
  - DMA Error Register (DMAERR), page 14-41
  - DMA Debug Event Status Register (DMADESR), page 14-43
  - DMA Local Profiling Configuration Register (DMALPCR), page 14-43
  - DMA Round-Robin Priority Group Update Register (DMARRPGUR), page 14-44
  - DMA Channel Active Status Register (DMACHASTR), page 14-45
  - DMA Channel Freeze Status Register (DMACHFSTR), page 14-45
- DMA Channel Buffer Descriptors
  - Buffer Attributes (BD_ATTR), page 14-49
  - Multi-Dimensional Buffer Attributes (BD_MD_ATTR), page 14-52

Note: The DMA controller registers use a base address of: 0xFFF10000.

Note: If you are using external peripherals, you must configure the DMA peripheral interface block by programming the GPIO block to enable the multiplexing of the handshaking signals and configure the associated channel information in the General Configuration Registers. See **Section 14.6.4**, *Using the DMA Peripheral Interface Block*, on page 14-26 for details.

14.7.1 DMA Buffer Descriptor Base Registers x (DMABDBRx)

DMABDBR[0–15] DMA Buffer Descriptor Base Registers 0–15 Offset 0x000 + x*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				BDTPTR											
Type	R/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**
	BDTPTR												DESO			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each DMABDBRx holds the user-programmable addresses of the BD tables for DMA channel x. There are two fields: BDTPTR that identifies the base address of the Buffer Descriptor Table and DESO field that identifies the location offset for the destination buffer within the table. All the channel properties should be programmed, including the relevant BD, before the channel is enabled. For more information on BD address calculation, see the discussion in **Section 14.7.22**, *DMA Channel Buffer Descriptors*, on page 14-45.

Table 14-15. DMABDBRx Field Descriptions

Bits	Reset	Description	Setting
— 31–28	0	Reserved. Write to zero for future compatibility.	
BDTPTR 27–4	0	**Buffer Descriptor Table Pointer** Holds the 24 most significant bits, out of the 32 bit Buffer Descriptors Table (BDT) address.	
DESO 3–0	0	**Destination Offset** Holds the offset of destination buffer descriptor table from the BTD base (BDTPTR × 256).	0000 Destination table offset is 0x20. 0001 Destination table offset is 0x40. 0010 Destination table offset is 0x80. 0011 Destination table offset is 0x100. 0100 Destination table offset is 0x200. 0101 Destination table offset is 0x400. 0110 Destination table offset is 0x600. 0111 Destination table offset is 0x800. 1000 Destination table offset is 0x1000. 1001 Destination table offset is 0x2000. 1010 Destination table offset is 0x3000. 1011 Destination table offset is 0x4000. 11xx Reserved.

14.7.2 DMA Controller Channel Configuration Registers x (DMACHCRx)

DMACHCR[0–15] DMA Controller Channel Configuration Registers 0–15 Offset 0x100 + x*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ACTV	SPRT	DPRT	SMDC	DMDC	—	SRCBDPT									
Type	R/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**
	RRPG			—	DPO	—	DESBDPT									
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMACHCR[0–15] configure the connection between a VCOP requestor and the corresponding VCOP channel. All the channel properties should be programmed, including the relevant BD, before a channel is enabled. The VCOP logic can modify some fields in these registers while the channel is active. To avoid conflict with the DMA logic, never write these registers while the respective channel is active.

Table 14-16. DMACHCRx Field Descriptions

Bits	Reset	Description	Settings
ACTV 31	0	**Active VCOP Channel** While a channel is disabled, all requests are ignored and any non-serviced request is lost. The DMA controller resets ACTV when the channel task completes. Never write a 0 to this bit when the channel is active. You must use DMACHDR to disable the channel first before disabling the channel. Disabling the channel does not reset its value immediately; the value is reset only when there are no more open requests on the bus interface. See also the DMA Channel Enable Register on page 14-31 and the DMA Channel Disable Register on page 14-32. **Written by:** User, DMA controller	0 Channel is disabled. 1 Channel is enabled.
SPRT 30	0	**Source Channel Port** Selects the MBus port associated with the source. **Written by:** User, DMA controller	0 Source is assigned to port 0 of the MBus interface. 1 Source is assigned to port 1 of the MBus interface.
DPRT 29	0	**Destination Channel Port** Selects the MBus interface associated with the destination. **Written by:** User, DMA controller	0 Destination is assigned to port 0 of the MBus interface. 1 Destination is assigned to port 1 of the MBus interface.
SMDC 28	0	**Source Multi-Dimensional Channel** The source can be either one-dimensional or multi-dimensional. **Written by:** User	0 Source is one-dimensional. 1 Source is multi-dimensional.
DMDC 27	0	**Destination Multi-Dimensional Channel** The destination can be either one-dimensional or multi-dimensional **Written by:** User	0 Destination is one-dimensional. 1 Destination is multi-dimensional.

Table 14-16. DMACHCRx Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 26	0	Reserved. Write to zero for future compatibility.	
SRCBDPT 25–16	0	**Source Buffer Pointer** BD number in the BD table assigned to the source. The maximum number of one-dimensional BDs per source is 1024, and the maximum number of multi-dimensional BDs per source is 512. For details on BD address calculation, see **Section 14.7.22**, *DMA Channel Buffer Descriptors*, on page 14-45. **Written by:** User, DMA controller	
RRPG 15–13	0	**Channel Round-Robin Priority Group** This field is valid only for round robin arbitration.See **Table 14-17** for selecting arbitration mode. For more details about round robin, see **14.3.1**, *Round-Robin Arbitration*. To update this field while the channel is active, use the DMA_RRPGUR register (see page 14-44) **Written by:** User, DMA controller	000 Highest priority. 011 Lowest priority. 1xx Reserved.
— 12	0	Reserved. Write to zero for future compatibility.	
DPO 11	0	Destination Port Optimize Specifies how the destination port is to be optimized. **Written by:** User	0 Optimize for destination port requests latency. 1 Optimize for destination port requests utilization.
— 10	0	Reserved. Write to zero for future compatibility.	
DESBDPT 9-0	0	**Destination Buffer Pointer** BD number in the BD table assigned to the destination. The maximum number of one-dimensional BDs per destination is 1024. The maximum number of multi-dimensional BDs per destination is 512. For details on BD address calculation, see **Section 14.7.22**. **Written by:** User, DMA controller	

14.7.3 DMA Controller Global Configuration Register (DMAGCR)

DMAGCR DMA Controller Global Configuration Register Offset 0x200

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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**
	—													PSCB	PSCA	AT
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAGCR defines DMA global parameters.

Table 14-17. DMAGCR Field Descriptions

Bits	Reset	Description	Register Setting
— 31–3	0	Reserved. Write to zero for future compatibility.	
PSCB 2	0	**Port 1 Slow Confirmation** For debug, the DMA supports slow confirmation for all transactions **Written by:** User	0 Fast confirmation on port 1, when possible. 1 Slow confirmation on port 1.
PSCA 1	0	**Port 0 Slow Confirmation** For debug, the DMA supports slow confirmation for all transactions **Written by:** User	0 Fast confirmation on port 0, when possible. 1 Slow confirmation on port 0.
AT 0	0	**Arbitration Type** Enables the DMA arbitration type. The DMA arbitration type is defined in the DMAGCR. See **Section 14.3**, *Arbitration Types,* on page 14-19 for details on arbitration. **Written by:** User	0 Enable round-robin arbitration. 1 Enable EDF arbitration.

14.7.4 DMA Channel Enable Register (DMACHER)

DMACHER DMA Channel Enable Register Offset 0x204

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	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**
	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Type	W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in DMACHER corresponds to DMACHCRx[ACTV]. When an ENx bit is set, it activates channel x. When ENx bit is cleared, it does not affect the channel. DMACHER is cleared at reset, and the user enables a channel request by setting the appropriate bit. The register allows simultaneous activation of channels after they are configured. While channel x is disabled, all requests are ignored and any non-serviced request is lost. The DMA controller logic resets the ENx bit when the channel task completes.

14.7.5 DMA Channel Disable Register (DMACHDR)

DMACHDR DMA Channel Disable Register Offset 0x20C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	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**
	DIS15	DIS14	DIS13	DIS12	DIS11	DIS10	DIS9	DIS8	DIS7	DIS6	DIS5	DIS4	DIS3	DIS2	DIS1	DIS0
Type	W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the DMACHDR corresponds to a channel. Writing a 1 to DISx disables channel x. Writing a 0 to DISx does not affect the channel. DMACHDR is cleared at reset. The register allows simultaneous deactivation of channels during normal operation. While channel x is disabled, all requests are ignored and any non-serviced request is lost. The DISx bit is reset by the DMA logic.

When the user writes either a 1 to DMACHDR[DISx] or a 0 to DMACHCRx[ACTV], the channel is shut down. If more channel requests are pending on the bus interface, the channel is not disabled. The DMACHDR[DISx] and corresponding DMACHER[ENx] and CHCRx[ACTV] are all set until the pending channel transactions are closed. When all transactions are closed, the DMA logic resets the DMACHER[ENx] and DMACHCRx[ACTV] bits. After the channel is disabled, you must poll DMACHASTR to acknowledge that the channel is disabled. The interrupt latency in the system must be considered as well. When you are sure that the channel is disabled and there is no previous pending interrupts, the channel can be activated.

14.7.6 DMA Channel Freeze Register (DMACHFR)

DMACHFR DMA Channel Freeze Register Offset 0x214

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D15	S15	D14	S14	D13	S13	D12	S12	D11	S11	D10	S10	D9	S9	D8	S8
Type	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**
	D7	S7	D6	S6	D5	S5	D4	S4	D3	S3	D2	S2	D1	S1	D0	S0
Type	W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Setting an Sx bit freezes the corresponding channel source. Setting a Dx bit freezes the corresponding channel destination. When a bit is set, the corresponding channel settings remain valid and new DREQ requests are considered, but the DMA controller does not issue any transactions to the specified channel. This register is write only; writing a 1 to a bit toggles its value (that is, if the value is 0, it sets the bit and if it is 1, it clears the bit). Writing a zero to the bits has no effect. The DMACHFR bits are all cleared by reset. Activating a channel clears the corresponding DMACHFR bits (Sx and Dx). The register allows simultaneous freezing of channels during normal operation

Note: The DMA channels do not freeze immediately; therefore, after a channel freeze is set, the DMA controller can issue new transactions for the channel until its pipeline is cleared.

Note: When the DMA channel becomes frozen, data may be left in the FIFO.

14.7.7 DMA Channel Defrost Register (DMACHDFR).

DMACHDFR DMA Channel Defrost Register Offset 0x224

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D15	S15	D14	S14	D13	S13	D12	S12	D11	S11	D10	S10	D9	S9	D8	S8
Type	W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	D7	S7	D6	S6	D5	S5	D4	S4	D3	S3	D2	S2	D1	S1	D0	S0
Type	W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Setting an Sx bit defrosts the corresponding channel source. Setting a Dx bit defrosts the corresponding channel destination. Defrosting a channel allows it to return to normal functioning. This register is write only; writing a 1 to a bit toggles its value (that is, if the value is 0, it sets the bit and if it is 1, it clears the bit). Writing a zero to the bits has no effect. The DMACHDFR bits

are all set by reset. Activating a channel sets the corresponding DMACHDFR bits (Sx and Dx). The register allows simultaneous defrosting of channels

14.7.8 DMA Time-To-Dead Line Registers x (DMAEDFTDLx)

DMAEDFTDL[0–15] DMA Time-To-Dead Line Registers 0–15 Offset 0x234 + x*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ENC	—							CURRENT_COUNT							
Type	R/W	R														
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**
	THRESHOLD								BASE_COUNT							
Type	R/W															
Reset	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1

Table 14-18 describes the fields of DMAEDFTDL[0–15].

Table 14-18. DMAEDFTDL[0–15] Field Descriptions

Bits	Reset	Write By	Description	Settings
ENC 31	0	User	**ENC** This field enable this counter. The counter will start counting when a task is asserted and this field set. **Note:** Enabling and disabling the channels change the counters value. Disabling channels stops the counting and enabling them reloads the counters with their base values.	0 Counter disabled 1 Counter enabled
— 30–24	0	Reserved. Write to zero for future compatibility.		
CURRENT_ COUNT 23–16	0	DMA	**Current Counter Value** This field holds the current value of the counter. Upon enabling the channel the counter is loaded with the base value. The counter counts down as long as the channel is enabled. The counter will stop counting when the channel is disabled or when counter reached zero and task is not completed yet. The counter resumes counting when the buffer ends according to the BD_ATTR[EDF].	
THRESHOLD 15–8	0x02	User	**Time to Dead Line Threshold** The threshold defines the value of the counter when the DMA task is due. The maximum threshold value is 0xff and the minimum is 2. **Note:** The DMA logic sets the priority of the channels according to counters threshold value.	
BASE_COUNT 7–0	0xFF	User	**Base Count Value** The base count value is set by the user before enabling the associated channel. When the DMA reinitializes the counter it reloads the counter with BASE_COUNT. The BASE_COUNT maximum value is 0xff and the minimum is 0. **Note:** Do not change the base count value of active channels.	

14.7.9 DMA EDF Control Register (DMAEDFCTRL)

DMAEDFCTRL DMA EDF Control Register Offset 0x334

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—														CLK_SRC	
Type	R/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
	CLK_DIV															
Type	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 14-19 describes the fields of the DMAEDFCTRL registers.

Table 14-19. DMAEDFCTRL Field Descriptions

Bits	Write by	Description	Setting
— 31–18	—	Reserved. Write to zero for future compatibility.	
CLK_SRC 17–16	User	**Clock Source** There are four clock sources for the EDF counters.	00: DMA clock divided by 16 01: Source 1 10: Source 2 11: Source 3
CLK_DIV 15–0	User	**Clock Divider** Divide the clock source (selected by CLK_SRC) for the EDF counters. The DMAEDFTDLx clock frequency is EDF clock (selected by CLK_SRC) divided by CLK_DIV. The maximum CLK_DIV value is 0xFFFF and the minimum is 1.	

14.7.10 DMA EDF Mask Register (DMAEDFMR)

DMAEDFMR DMA EDF Mask Register Offset 0x338

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0
Type	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the DMAEDFMR corresponds to an interrupt request bit in the DMAEDFSTR. When a bit is set, it enables the generation of an interrupt request of the corresponding counter.

DMAEDFMR is cleared at reset, and the user enables a counter interrupt request by setting the appropriate bit.**Table 14-20** describes the fields of the DMAEDFMR.

Table 14-20. DMAEDFMR Field Descriptions

Bits	Write by	Description	Setting
— 31–16	—	Reserved. Write to zero for future compatibility.	
M[15–0] 15–0	User	**Masks 15–0** Each bit corresponds to an interrupt request bit in the DMAEDFSTR. Setting the bit enables generation of the respective interrupt request.	0 Interrupt masked. 1 Interrupt enabled.

14.7.11 DMA EDF Mask Update Register (DMAEDFMUR)

DMAEDFMUR DMA EDF Mask Update Register Offset 0x340

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MASK_CH3						NM3	EN3	MASK_CH2						NM2	EN2
Type	R/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
	MASK_CH1						NM1	EN1	MASK_CH0						NM0	EN0
	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DMAEDFMUR enables modifying the DMAEDFMR registers without a read modify write operation. You can modify up to four channels with one action. **Table 14-21** describes the fields of the DMAEDFMUR.

Table 14-21. DMAEDFMUR Field Descriptions

Bits	Write by	Description	Setting
MASK_CH3 31–26	User	**Channel Number** Indicate the channel number of the DMAEDFMR to change.	000000–001111 Channel number. 01xxxx Reserved. 1xxxxx Reserved.
NM3 25	User	**New Channel Mask Value** Stores the new value of DMAEDFMR[MASK_CH3].	0 Not masked. 1 Masked.
EN3 24	User	**Enable Mask/Unmask Updating** When set, updates DMAEDFMR[MASK_CH3] according to NM3. When DMAEDFMR[MASK_CH3] is updated, the DMA controller clears this bit.	0 Update occurred. 1 Perform update.
MASK_CH2 23–18	User	**Channel Number** Indicate the channel number of the DMAEDFMR to change.	000000–001111 Channel number. 01xxxx Reserved. 1xxxxx Reserved.
NM2 17	User	**New Channel Mask Value** Stores the new value of DMAEDFMR[MASK_CH2].	0 Not masked. 1 Masked.

Table 14-21. DMAEDFMUR Field Descriptions (Continued) (Continued)

Bits	Write by	Description	Setting
EN2 16	User	**Enable Mask/Unmask Updating** When set, updates DMAEDFMR[MASK_CH2] according to NM3. When DMAEDFMR[MASK_CH2] is updated, the DMA controller clears this bit.	0 Update occurred. 1 Perform update.
MASK_CH1 15–10	User	**Channel Number** Indicate the channel number of the DMAEDFMR to change.	000000–001111 Channel number. 01xxxx Reserved. 1xxxxx Reserved.
NM1 9	User	**New Channel Mask Value** Stores the new value of DMAEDFMR[MASK_CH1].	0 Not masked. 1 Masked.
EN1 8	User	**Enable Mask/Unmask Updating** When set, updates DMAEDFMR[MASK_CH1] according to NM3. When DMAEDFMR[MASK_CH1] is updated, the DMA controller clears this bit.	0 Update occurred. 1 Perform update.
MASK_CH0 7–2	User	**Channel Number** Indicate the channel number of the DMAEDFMR to change.	000000–001111 Channel number. 01xxxx Reserved. 1xxxxx Reserved.
NM0 1	User	**New Channel Mask Value** Stores the new value of DMAEDFMR[MASK_CH0].	0 Not masked. 1 Masked.
EN0 0	User	**Enable Mask/Unmask Updating** When set, updates DMAEDFMR[MASK_CH0 according to NM3. When DMAEDFMR[MASK_CH0] is updated, the DMA controller clears this bit.	0 Update occurred. 1 Perform update.

14.7.12 DMA EDF Status Register (DMAEDFSTR)

DMAEDFSTR DMA EDF Status Register Offset 0x344

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
	W1C															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the DMAEDFSTR corresponds to an EDF threshold violation status the corresponding channel. If set, a bit associated with a channel indicates that EDF threshold violation occurred for that channel. A bit is cleared by writing one to it. Writing zero does not affect a bit value. It is possible to clear several bits at a time. **Table 14-22** describes the fields of the DMAEDFSTR.

Table 14-22. DMAEDFSTR Field Descriptions

Bits	Write by	Description	Setting
— 31–16	—	Reserved. Write to zero for future compatibility.	
I[15–0] 15–0	User	**Interrupt for Threshold Violation 15–0** Each bit corresponds to an interrupt request bit in the DMAEDFSTR. Setting the bit enables generation of the respective interrupt request.	0 No interrupt. 1 Interrupt request issued.

14.7.13 DMA Mask Register (DMAMR)

DMAMR DMA Mask Register Offset 0x34C

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	D15	S15	D14	S14	D13	S13	D12	S12	D11	S11	D10	S10	D9	S9	D8	S8
Type	R/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**
	D7	S7	D6	S6	D5	S5	D4	S4	D3	S3	D2	S2	D1	S1	D0	S0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the DMAMR corresponds to an interrupt request bit in the DMASTR. When a bit is set, it enables the generation of an interrupt request on the corresponding interrupt line. DMAMR is cleared at reset, and you enable a channel interrupt request by setting the appropriate bit.

Note: Setting the Sn bits has no effect because in the MSC8156E, only unidirectional destination channel interrupts are implemented as described in **Section 14.4.1**.

14.7.14 DMA Mask Update Register (DMAMUR)

DMAMUR DMA Mask Update Register Offset 0x35C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MASKCH3					DES3	NM3	EN3	MASKCH2					DES2	NM2	EM2
Type	R															
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**
	MASKCH1					DES1	NM1	EN1	MASKCH0					DES0	NM0	EN0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAMUR is a special register that allows you to modify the DMAMR registers without a read-modify-write operation.

Table 14-23. DMAMUR Field Descriptions

Bits	Reset	Description	Settings
MASKCH3 31–27	0	**Channel Number** The channel number to which DMAMR should be changed. **Written by:** User	00000–01111: Channel number. 1xxxx Reserved.
DES3 26	0	**Channel Number Destination** Destination channel number to which DMAMR should be changed. **Written by:** User	0 Channel number source. 1 Channel number destination.
NM3 25	0	**New Channel Mask value** The new value of DMAMR[MASKCH3, DES3]. **Written by:** User	0 Unmask. 1 Mask.
EN3 24	0	**Enable MASK/UNMASK Update** Updates DMAMR[MASK_CH3, DES3] according to NM3. Then the DMA controller clears this bit. **Written by:** User, DMA controller	
MASKCH2 23–19	0	**Channel Number** The channel number to which DMAMR should be changed. **Written by:** User	00000–01111: Channel number. 1xxxx: Reserved.
DES2 18	0	**Channel Number Destination** Destination channel number to which DMAMR should be changed. **Written by:** User	0 Channel number source. 1 Channel number destination.
NM2 17	0	**New Channel Mask value** The new value of DMAMR[MASKCH3, DES2]. **Written by:** User	0 Unmask. 1 Mask.
EN2 16	0	**Enable MASK/UNMASK Update** Updates the DMAMR[MASKCH2, DES2] according to NM2. Then the DMA controller clears this bit. **Written by:** User, DMA controller	
MASKCH1 15–11	0	**Channel Number** The channel number to which DMAMR should be changed. **Written by:** User	00000–01111: Channel number. 1xxxx: Reserved

Table 14-23. DMAMUR Field Descriptions (Continued)

Bits	Reset	Description	Settings
DES1 10	0	**Channel Number Destination** Destination channel number to which DMAMR should be changed. **Written by:** User	0 Channel number source. 1 Channel Number destination.
NM1 9	0	**New Channel Mask value** The new value of DMA_MR[MASK_CH1, DES1]. **Written by:** User	0 Unmask. 1 Mask.
EN1 8	0	**Enable MASK/UNMASK Update** Updates DMAMR[MASKCH0, DES0] according to NM0. Then the DMA controller clears this bit. **Written by:** User, DMA controller	
MASKCH0 7–3	0	**Channel Number** The channel number to which DMAMR should be changed. **Written by:** User	00000–01111: Channel number. 1xxxx Reserved
DES0 2	0	**Channel Number Destination** Destination channel number to which DMAMR should be changed. **Written by:** User	0 Channel number source. 1 Channel number destination.
NM0 1	0	**New Channel Mask value** The new value of DMAMR[MASKCH0, DES0]. **Written by:** User	0 Unmask. 1 Mask.
EN0 0	0	**Enable MASK/UNMASK Update** Updates DMAMR[MASKCH0, DES0] according to NM0. Then the DMA controller clears this bit. **Written by:** User, DMA controller	

14.7.15 DMA Status Register (DMASTR)

DMASTR DMA Status Register Offset 0x360

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D15	S15	D14	S14	D13	S13	D12	S12	D11	S11	D10	S10	D9	S9	D8	S8
Type	R															
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**
	D7	S7	D6	S6	D5	S5	D4	S4	D3	S3	D2	S2	D1	S1	D0	S0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the DMASTR corresponds to a channel. The source status for each channel is controlled in the BD associated with the channel. If set, a bit associated with a channel indicates that the buffer ends when BD_ATTR[SST] is set or it is the last buffer. A bit is cleared by writing a value of one to it. Writing zero does not affect a bit value. Several bits can be cleared at one time.

Note: You must clear the Dx and Sx bits before enabling the channel.

14.7.16 DMA Error Register (DMAERR)

DMAERR DMA Error Register Offset 0x370

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BDSZ	PACH						PADEST	—	PBCH						PBDEST
Type	R	R/W							R	R/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**
	PAE	PBE	—	THV	PRTYP	PRTYF	PRTYB	PRTY	—	PRTYCH						PRTYD
Type	R/W		R													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAERR holds the source for the error interrupt. The error interrupt output is unmasked in the DMA controller. A bit is cleared by writing a value of one to it. Writing zero does not affect a bit value. Several bits can be cleared at one time. If a port error occurs, all channels assigned to the port enter a freeze state. You must reprogram the channel that caused the error and reactivate it. You may decide to defrost other port assigned channels and continue normally. For a BD size error or parity error, only the channel that caused the error is frozen.

Table 14-24. DMAERR Description

Bits	Reset	Description	Settings
BDSZ 31	0	**Buffer Descriptor Size** Indicates whether the buffer descriptor size is programmed to zero. See also **Table 14-28** and **Table 14-30**.	0 No BD_SIZE or MD_BD_SIZE of 0 detected. 1 BD_SIZE or MD_BD_SIZE of 0 detected.
PACH 30–25	0	**First Port 0 Channel to Cause Bus Error** Indicates which channel caused the first error on bus interface port A.	000000 - 001111: channel number 01xxxx: Reserved 10xxxx: Reserved
PADEST 24	0	**Error of Port 0 Destination Channel** Indicates whether the last error on port 0 was caused by channel source or destination.	0: Source transaction error. 1: Destination transaction error.
— 23	0	Reserved. Write to zero for future compatibility.	
PBCH 22–17	0	**First Port 1 Channel to Cause Bus Error** Indicates which channel caused the first error on bus interface port B.	000000–001111: channel number. 01xxxx Reserved 10xxxx: Reserved
PBDEST 16	0	**Error of Port 1 Destination Channel** Indicates whether the last error on port 1 was caused by a channel source or destination.	0 Source transaction error. 1 Destination transaction error.
PAE 15	0	**Port 0 Transfer Error Indication** Indicates whether there is an acknowledged transfer error on port 0.	0 No transfer error acknowledged on port 0. 1 Transfer error acknowledged on port 0.
PBE 14	0	**Port 1 Transfer Error Indication** Indicates whether there is an acknowledged transfer error on port 1.	0 No transfer error acknowledged on port 1. 1 Transfer error acknowledged on port 1.
— 13	0	**Reserved**, write zero for future compatibility.	

Table 14-24. DMAERR Description (Continued)

Bits	Reset	Description	Settings
THV 12	0	**Threshold Violation** The channel that violated the deadline is indicated in the DMA Counter Status Register (DMACNTSTR). THV is set as long as there is a set bit in DMACNTSTR. THV is automatically cleared when all bits in DMACNTSTR are cleared.	0 No deadline violation. 1 Deadline violation.
PRTYP 11	0	**Parity Error on PRAM** Indicates that the parity error occurred in the PRAM.	0 No error indication on the PRAM. 1 PRAM parity error indication.
PRTYF 10	0	**Parity Error on FIFOs** Indicates that the parity error occurred in the FIFOs.	0 No error indication in the FIFO's. 1 FIFO's parity error indication.
PRTYB 9	0	**Parity Error on Bus interface** Indicates that the parity error occurred in the BI.	0 No error indication in the BI. 1 BI parity error indication.
PRTY 8	0	**Parity Error** Indicates any parity error.	0 No error indication. 1 Error indication.
— 7	0	**Reserved**, write zero for future compatibility.	
PRTYCH 6–1	0	**Parity Channel** First channel that caused parity error Indicates by which channel the last parity error was caused.	000000–001111: Channel number. 01xxxx Reserved. 10xxxx Reserved.
PRTYD 0	0	**Parity Error Destination** Indicates whether the first parity error was caused by a channel source or destination.	0 Source transaction error. 1 Destination transaction error.

14.7.17 DMA Debug Event Status Register (DMADESR)

DMADESR DMA Debug Event Status Register Offset 0x374

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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**
	—														EXT	DBG
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DMADESCR reflects whether an external debug request was received and whether the DMA is in Debug mode. **Table 14-25** describes the fields of the DMADESR.

Table 14-25. DMADESR Field Descriptions

Bits	Write By	Description	Settings
— 31–2	—	Reserved. Write to zero for future compatibility.	
EXT 1	DMA	**External DMA Debug Request Event** Set when external debug event occurs.	0 Normal operation 1 External DMA debug request
DBG 0	DMA	**Debug Mode Status** Set when the DMA is in full Debug mode.	0 Normal operation. 1 DMA in Debug mode.

14.7.18 DMA Local Profiling Configuration Register (DMALPCR)

DMALPCR DMA Local Profiling Configuration Register Offset 0x378

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—									CHAPRO						DEST
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMALPCR selects the channel for system profiling.

Table 14-26. DMALPCR Field Descriptions

Bits	Reset	Description	Settings
— 31–7	0	Reserved. Write to zero for future compatibility.	

Table 14-26. DMALPCR Field Descriptions (Continued)

Bits	Reset	Description	Settings
CHAPRO 6–1	0	**Channel Profiled** Selects the channel to be profiled. **Write by:** User	000000–001111: Channel number. 01xxxx Reserved. 10xxxx Reserved.
DEST 0	0	**Destination Channel Profiled** Specifies whether source or destination requests are profiled. **Write by:** User	0 Channel source requests are profiled. 1 Channel destination requests are profiled.

14.7.19 DMA Round-Robin Priority Group Update Register (DMARRPGUR)

DMARRPGUR DMA Round-Robin Priority Group Update Register Offset 0x37C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—						CH						NRRPG			EN
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMARRPGR is a special register that allows you to modify the DMACHCR[RRPG] bit without a read-modify-write operation.

Note: Do not modify this register while the DMA controller is in EDF mode (DMAGCR[AT] is set—see page 14-31).

Table 14-27. DMARRPGUR Field Descriptions

Bits	Description	Settings
— 31–10	Reserved. Write to zero for future compatibility.	
CH 9–4	**Channel Number** The channel number to which DMACHCR[RRPG] should be changed.	000000–001111: Channel number. 01xxxx Reserved. 1xxxxx Reserved.
NRRPG 3–1	**New Channel Round-Robin Priority Group** The new value of RRPG to be written to the corresponding CHCR of the channel.	000 Highest priority. ... 011 Lowest priority. 1xx Reserved.
EN 0	**Enable RRPG Update** Enables the RRPG update. Then the DMA controller clears this bit	

14.7.20 DMA Channel Active Status Register (DMACHASTR)

DMACHASTR DMA Channel Active Status Register Offset 0x380

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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**
	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each bit in the DMACHASTR corresponds to the active status of the associated channel. If set, a bit associated with a channel indicates that the channel is still active. A channel can stay active even after DMACHCR[ACTV] is cleared or DMACHDR[DISx] is set. A DMACHASTR bit is reset only when its corresponding channel completes the shutdown procedure.

14.7.21 DMA Channel Freeze Status Register (DMACHFSTR)

DMACHFSTR DMA Channel Freeze Status Register Offset 0x388

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D15	S15	D14	S14	D13	S13	D12	S12	D11	S11	D10	S10	D9	S9	D8	S8
Type	R															
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**
	D7	S7	D6	S6	D5	S5	D4	S4	D3	S3	D2	S2	D1	S1	D0	S0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

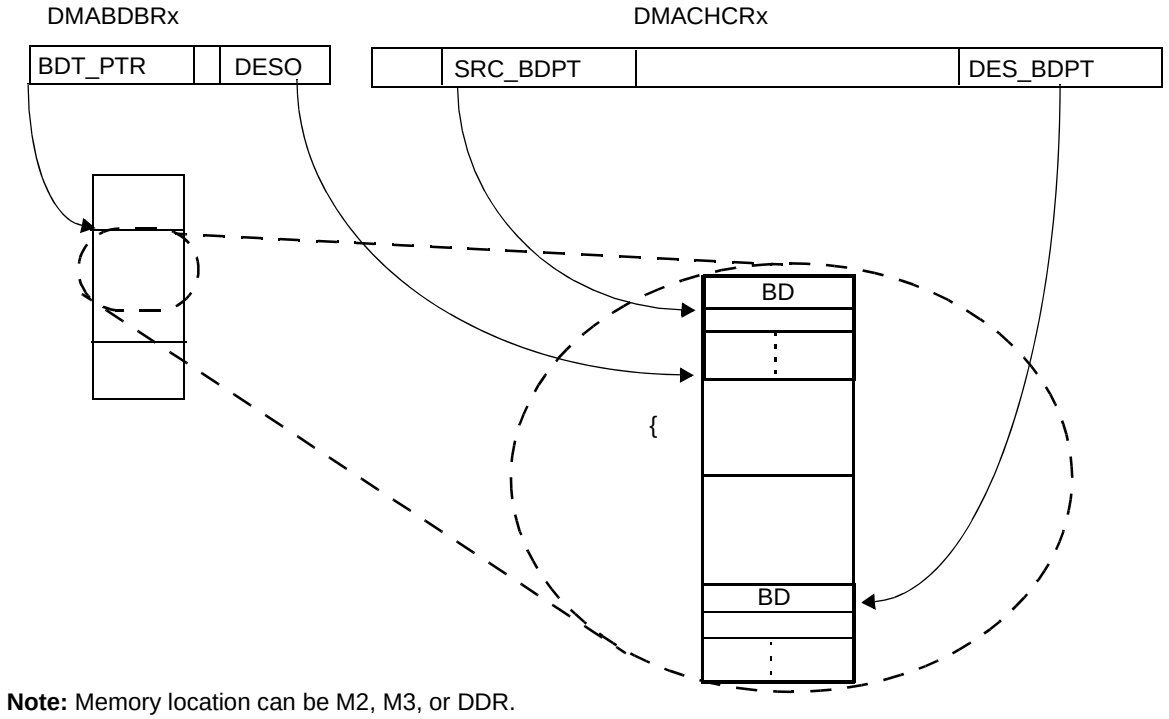
Each bit in the DMACHFSTR corresponds to the freeze status of the associated channel. If set, a bit associated with a channel indicates that the channel is still frozen.

Note: The corresponding bits are cleared when a channel is activated.

Note: When a bit is set as a result of BD_ATTR[FRZ] or BD_MD_ATTR[FRZ], it means that the DMA internal logic has finished serving the current BD and will not fetch or process the next BD until the channel is unfrozen. It does not imply that the current BD data or update has reached its destination. To guarantee that data or an updated BD is written to memory, use the interrupt mechanism or poll the channel status register.

14.7.22 DMA Channel Buffer Descriptors

Figure 14-13 shows a diagram of the BD pointer scheme.



Note: Memory location can be M2, M3, or DDR.

Figure 14-13. Buffer Descriptor Pointers

Following are the actual addresses of the source and destination BDs for any channel:

- BDT_BASE = DMABDBR[BDT_PTR] × 256
- Source BD = BDT_BASE + DMACHCR[SRCBDPT] × 16 × (DMACHCR[SMDC] + 1)
- Destination BD: BDT_BASE + offset + DMACHCR[DESBDPT] × 16 × (DMACHCR[DMDC] + 1)
- Offset is the decoded value of DMABDBR[DESO]

The VCOP channel BDs are located in memory outside the VCOP. Each channel has a BD table to hold the BDs for both source and destination buffers. All BDs of all channels must be located in memory connected to MBus interface 0. **Figure 14-14** shows the structure of one-dimensional BD, which is a 128-bit entry.

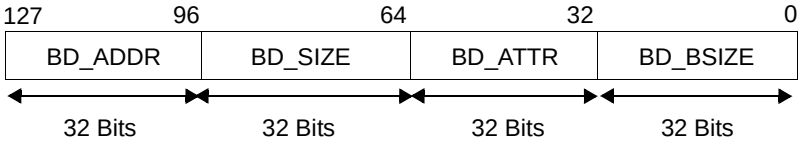


Figure 14-14. DMA Channel BD One-Dimensional Line

Figure 14-15 shows the structure of a multi-dimensional BD, which is a 256-bit entry.

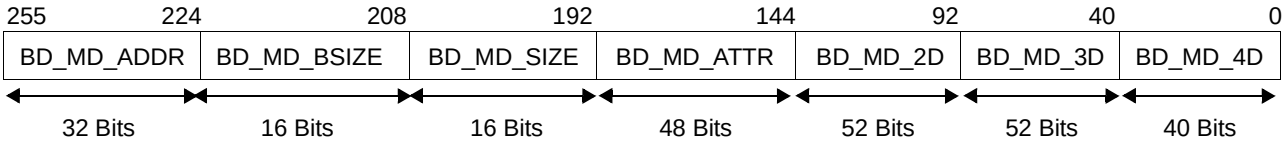


Figure 14-15. DMA Channel Multi-Dimensional Buffer Descriptor

Table 14-16 shows an example structure for a BD table that holds multi-dimensional read and one-dimensional DMA write tasks. The read channel uses 512 BDs of multi-dimensional BDs, which is the maximum available for multiple dimensions. The write channel uses 1024 one-dimensional BDs, which is the maximum available for one dimension. For details on BD address calculation.

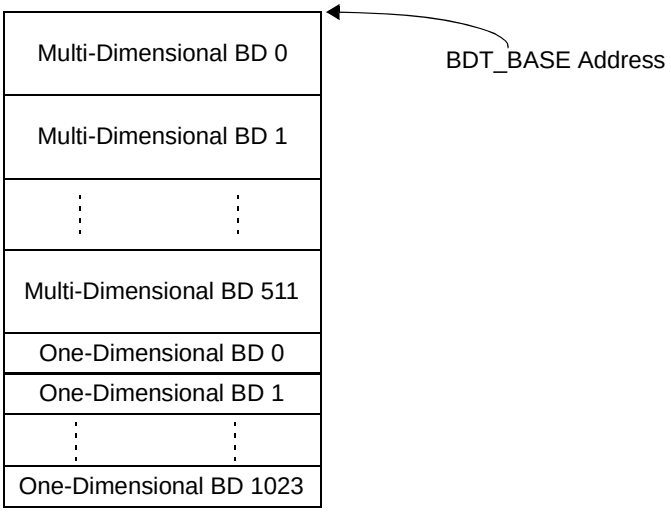


Figure 14-16. Example BD Table with a Mixed Dimensional Structure

The BDs are either one-dimensional or multi-dimensional. One-dimensional BDs are chained only to one-dimensional BDs and multi-dimensional BDs are chained only to multi-dimensional BDs. The types of source and destination BDs are defined in the DMACHCR (see page 14-29). **Table 14-28** lists the channel parameters for a one-dimensional BD.

Table 14-28. One-Dimensional BD Field Descriptions

Bits	Description
BD_ADDR 127–96	**Current Buffer Address** Holds the buffer address pointer. This value increments on every transaction the DMA controller issues for this buffer. If the buffer is cyclic, the original address value is restored when the BD_SIZE value reaches zero. For details, refer to **Section 14.2.2**, *One-Dimensional Cyclic Buffer,* on page 14-5.
BD_SIZE 95–64	**Size of Transfer Left for Current Buffer** Contains the remaining size of the buffer. This value decrements by the transfer size each time the DMA controller issues a transaction until it reaches zero. When BD_SIZE reaches zero, the value is restored to the value of BD_BSIZE. **Note:** The BD_SIZE must not be programmed as zero. A value of zero sets DMAERR[BDSZ] and freezes the channel. To reactivate the channel, you must disable the channel, reprogram the BDs, and reactivate the channel.
BD_ATTR 63–32	**Buffer Attributes** This 32-bit parameter describes the attributes of the channel handling this buffer. The fields of the BD_ATTR parameter are described in **Table 14-29**.
BD_BSIZE 31–0	**Buffer Base Size** Holds the base size of the buffer.

14.7.22.1 Buffer Attributes (BD_ATTR)

BD_ATTR Buffer Attributes

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SST	CYC	CONT	NPRT	NO_INC	—	NBD									
Type	R/W															
Reset	Undefined															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CNT		PP		TSZ				—	FRZ	MR	—		BTSZ		
Type	R/W															
Reset	Undefined															

Table 14-29. BD_ATTR Field Descriptions

Bits	Description	Settings
SST 31	**Set Status** Indicates whether to set the associated status bit in DMASTR when size reaches zero and the last data transaction ends. Setting this bit in the destination buffer issues a masked interrupt request.	0 Do not set status 1 Set status.
CYC 30	**Cyclic Address** Indicates the behavior of BD_ADDR when BD_SIZE reaches zero. For details on cyclic buffers, see **Section 14.2**, *Buffer Types,* on page 14-2.	0 Sequential address: BD_ADDR increments. 1 Cyclic address: BD_ADDR is restored to the original value for a one-dimensional buffer.
CONT 29	**Continuous Buffer Mode** Indicates whether buffer is to close when BD_SIZE reaches zero. **Note:** Unlike multidimensional mode, there is no *last buffer indicator* in simple mode. If your application is chaining multiple single dimension buffers, you must set this bit for all but the last BD in the chain. The cleared CONT bit in the last BD effectively becomes the last buffer indicator for the chain.	0 Buffer closes. 1 Buffer continues operating.
NPRT 28	**Next Port** When size reaches zero and CONT is set, DMACHCR[SPRT] (for source buffers) or DMACHCR[DPRT] (for destination buffers) is updated according to the NPRT field:	0 Next buffer port is MBus port 0. 1 Next buffer port is MBus port 1.
NO_INC 27	**Increment Address** Indicates the behavior of the buffer address after the request is serviced. **Note:** When this bit is set, the DMA controller always issues requests for the same address. It aligns (to the transfer size) on the first transfer as it does for any other first transaction. The DMA controller handles BD_SIZE as if it were a normal buffer, that is, it issues requests with the total byte count size as recorded in the BD_SIZE field.	0 Increment address. 1 Do not increment address.
— 26	Reserved. Write to zero for future compatibility.	
NBD 25–16	**Next Buffer** When size reaches zero and CONT is set, the next request calls the buffer to which NBD points.	

Table 14-29. BD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
CNT 15–14	**Channel Counter** This field is valid only when the arbitration is time-based. It characterizes the time-based arbitration mechanism for continuous buffers when the buffer size reaches zero and applies only when switching buffers.	00 Continuous: Channel and counter continue working normally. 01 Reserved. 10 Reserved. 11 Resets the channel counter.
PP 13–12	**Port priority** Define priority for this buffer. The bus Interface will set priority for this buffer. Note that the user must map the priority in system level.	00 Priority 0 (lowest). 11 Priority 3 (highest).
TSZ 11–8	**Transfer size** Indicates the maximum transfer size that the DMA will issue when request is detected.	0000 512 bytes 0001 1 byte. 0010 2 bytes. 0011 4 bytes. 0100 8 bytes. 0101 16 bytes. 0110 32 bytes. 0111 64 bytes. 1000 128 bytes. 1001 256 bytes. 1010 512 bytes. 1011 1024 bytes. 11xx Reserved.
— 7	Reserved. Write to zero for future compatibility.	
FRZ 6	**Freeze channel** When size reaches zero, the channel can be frozen. The already serviced requests continue normally. No further requests are issued for the associated channel until the host defrosts it.	0 Normal operation 1 Freeze channel.
MR 5	**Mask Requests Until Data Reached Destination** Indicates the behavior of the logic when BD_SIZE reaches zero. Typically, in continuous buffers, the channel should not be masked. However, there is an automatic mask when continuous buffers switch between ports. The DMA controller unmasks the requests when the last data reaches the destination.	0 Normal operation. 1 Mask requests until data reached destination.
— 4–3	Reserved. Write to zero for future compatibility.	
BTSZ 2–0	**Basic Transfer Size** The basic transfer size issued for the request. If BTSZ is greater than TSZ, the DMA controller uses TSZ for the transfer size.	000 64 bytes 001 1 byte 010 2 bytes 011 4 bytes 100 8 bytes 101 16 bytes 110 32 bytes 111 64 bytes

Table 14-30 shows the DMA channel parameters for a multi-dimensional buffer.

Table 14-30. Multi-Dimensional BD Field Descriptions

Bits	Description
BD_MD_ADDR 255–224	**Current Buffer Address** Holds the buffer address pointer. This value increments on every transaction the DMA controller issues for this buffer. When BD_MD_SIZE reached zero and the next dimension is active, the next dimension offset is added to the BD_MD_ADDR. For details, see **Section 14.2**, *Buffer Types,* on page 14-2.
BD_MD_BSIZE 223–208	**First Dimension Buffer Base Size** Contains the base size for the buffer first dimension.
BD_MD_SIZE 207–192	**First Dimension Buffer Size** Holds the remaining size for the buffer first dimension. This value is incremented by the transfer size each time the DMA controller issues a transaction, until it reaches zero. When BD_MD_SIZE reaches zero, its value is restored to the value of BD_MD_BSIZE. **Note:** BD_MD_SIZE must not be programmed to zero. Programming the value to zero sets DMAERR[BDSZ] and freezes the channel. To reactivate the channel, you must disable the channel, wait for the active status bit to go to clear, reprogram the BDs, and reactivate the channel.
BD_MD_ATTR 191–144	**Multi-Dimensional Buffer Attributes** This 48-bit parameter describes the multi-dimensional attributes of the channel handling this buffer. The BD_MD_ATTR parameters are described in **Table 14-31**.
BD_MD_2D 143–92	**Two-Dimensional Buffer Offset, Bcount, and Count** This 52-bit parameter holds the two-dimensional parameters of the channel handling this buffer. It holds the base count value, current count, and address offset. The Multi Dimension fields are described in **Table 14-32, *BD_MD_2D Field Descriptions,*** on page 14-55.
BD_MD_3D 91–40	**Three-Dimensional Buffer Offset, Bcount, and Count** This 52-bit parameter holds the three-dimension parameters of the channel handling this buffer. It holds the base count value, current count, and address offset. The Multi Dimension fields are described in **Table 14-33, *BD_MD_3D Field Descriptions,*** on page 14-55.
BD_MD_4D 39–0	**Four-Dimensional Buffer Offset and Count** This 40-bit parameter holds the four-dimensional parameters of the channel handling this buffer. It holds the base count value and current count. The multi-dimensional fields are described in **Table 14-34, *BD_MD_4D Field Descriptions,*** on page 14-55.

14.7.22.2 Multi-Dimensional Buffer Attributes (BD_MD_ATTR)

BD_MD_ATTR Multi-Dimensional Buffer Attributes

Bit	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
	SST	CYC	CONT	NPRT	NO_INC	—	NBD									
Type	R/W															
Reset	Undefined															

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CNT		PP		TSZ				—	FRZ	MR	—		BTSZ		
Type	R/W															
Reset	Undefined															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—					LAST	BD		SSTD		FRZD		CONTD		MRD	
Type	R/W															
Reset	Undefined															

Table 14-31. BD_MD_ATTR Field Descriptions

Bits	Description	Settings
SST 47	**Set Status** Indicates whether to set the associated status bit in DMASTR when size reaches zero and the last data transaction ends. Setting this bit in the destination buffer issues a masked interrupt request. See also the SSTD bit.	0 Do not set status. 1 Set status when the size of the dimension selected by SSTD reaches zero.
CYC 46	**Cyclic Address** Indicates the behavior of BD_MD_ADDR when BD dimension count reaches zero. For details on cyclic buffers, see **Section 14.2**, *Buffer Types,* on page 14-2. **Notes:** **1.** This field must not be set for four dimensional buffers. **2.** This field is not valid for BD_MD_ATTR[CONTD]<BD_MD+ATTTR[BD].	0 Sequential address: BD_MD_ADDR is incremented. 1 Cyclic address: the next dimension offset is added to BD_MD_ADDR.
CONT 45	**Continuous Buffer Mode** Specifies whether the buffer closes when CONTD dimension count reaches zero.	0 Buffer closes. 1 Buffer continues operating.
NPRT 44	**Next Port** When size reaches zero and CONT is set and the CONTD dimension count reaches zero, DMACHCR[SPRT] (for source buffers) or DMACHCR[DPRT] (for destination buffers) is updated according to the NPRT field.	0 Next buffer port is MBus port 0. 1 Next buffer port is MBus port 1.
NO_INC 43	**Increment Address** Indicates the behavior of the buffer address after a request is serviced. When a dimension is processed, the DMA adds the next dimension offset to the address, regardless of the status of NO_INC.	0 Increment address. 1 Do not increment address.
— 42	Reserved. Write to zero for future compatibility.	

Table 14-31. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
NBD 41–32	**Next Buffer** When size reaches zero, CONT is set, and the CONTD dimension count reaches zero, the next request calls the buffer to which NBD points. **Note:** For four dimensional buffers, if CONT is set, NBD must be different from the current BD.	
CNT 31–30	**Channel Counter** This field is valid only for a destination buffer only when the arbitration is time-based. It characterizes the time-based arbitration mechanism for continuous buffers when the buffer size reaches zero and applies only when switching buffers.	00 Continuous: Channel and counter continue working normally. 01 Reserved. 10 Reserved. 11 Resets the channel counter.
PP 29–28	**Port Priority** Defines the priority for the designated buffer. The bus interface sets the priority for this buffer. You must map the priority at the system level.	00 Priority 0 (lowest.). 11 Priority 3 (highest).
TSZ 27–24	**Transfer Size** Indicates the maximum transfer size that the DMA controller issues when a request is detected.	0000 512 bytes. 0001 1 byte. 0010 2 bytes. 0011 4 bytes. 0100 8 bytes. 0101 16 bytes. 0110 32 bytes. 0111 64 bytes. 1000 128 bytes. 1001 256 bytes. 1010 512 bytes. 1011 1024 bytes. 11xx Reserved.
— 23	Reserved. Write to zero for future compatibility.	
FRZ 22	**Freeze Channel** When size reached zero the channel can be freeze. The already serviced requests continue normally. No further requests are issued for the associated channel until the host defrost it. See also the FRZD field.	0 Normal operation. 1 Freeze channel when size reaches zero on the dimension selected by FRZD.
MR 21	**Mask Requests Until Data Reached Destination** Indicates the behavior of the logic when BD_MD_SIZE reaches zero. In continuous buffers, the channel is usually not masked. There is an automatic mask when ports are switched in a continuous buffer. The DMA controller unmasks the requests when last data reaches the destination. See also the MRD field.	0 Normal operation. 1 Mask requests until data reached destination on the dimension selected by MRD.
— 20–19	Reserved. Write to zero for future compatibility.	

Table 14-31. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
BTSZ 18–16	**Basic Transfer Size** The basic transfer size issued by the request. If BTSZ is greater than TSZ, the DMA controller uses the TSZ value.	000 64 bytes. 001 1 byte. 010 2 bytes. 011 4 bytes. 100 8 bytes. 101 16 bytes. 110 32 bytes. 111 64 bytes.
— 15–11	Reserved. Write to zero for future compatibility.	
LAST 10	**Last Buffer In Chain** Set this bit only to avoid an endless task condition when CONT is set and CONTD is smaller than BD.	0 Not the last buffer in the chain. 1 Last buffer in the chain.
BD 9–8	**Buffer Dimension** Indicates the dimension of the buffer.	00 1 dimension. 01 2 dimensions. 10 3 dimensions. 11 4 dimensions.
SSTD 7–6	**Set Status Dimension** When BD_MD_SIZE reaches zero and BD_MD_ATTR[SST] = 1, the status bit is set for the channel in DMASTR. For details, see page 14-40. The SSTD field defines the dimension on which the status bit is set.	00 BD_MD_SIZE reached zero, 1D. 01 M2D_COUNT reached zero, 2D. 10 M3D_COUNT reached zero, 3D. 11 M4D_COUNT reached zero, 4D.
FRZD 5–4	**Freeze Dimension** When the selected dimension is processed, the internal logic masks all channel requests and freezes the channel. The host must defrost the channel for further service.This field is valid only if FRZ is set in the BD_MD_ATTR.	00 Mask and freeze channel when first dimension ends. 01 Mask and freeze channel when second dimension ends. 10 Mask and freeze channel when third dimension ends. 11 Mask and freeze channel when fourth dimension ends.
CONTD 3–2	**Continuous Buffer Mode Dimension Select** Indicates the dimension after which the channel switches to the next multi-dimensional BD. This field is valid only if CONT is set in the BD_MD_ATTR. **Notes:** **1.** The value must not be greater than BD. **2.** When NBD is equal to the current BD and CONT is set, CONTD must equal BD. **3.** If CONT is set and CONTD < BD, the BD area is updated by the DMA controller. Do not access this area until the DMA task is completed. **4.** If CONT is set, CONTD < BD, and NPRT is different from the current port, MR must be set an MRD must be equal to CONTD.	00 Switch to next BD when BD_MD_SIZE reaches zero, 1D 01 Switch to next BD when M2D_COUNT reaches zero, 2D. 10 Switch to next BD when M3D_COUNT reaches zero, 3D. 11 Switch to next BD when M4D_COUNT reaches zero, 4D.

Table 14-31. BD_MD_ATTR Field Descriptions (Continued)

Bits	Description	Settings
MRD 1–0	**Mask Requests Dimension** Indicates the dimension after which the channel masks requests until the data reaches its destination. This field is valid only if MR is set in the BD_MD_ATTR.	00 Mask requests when BD_MD_SIZE reaches zero, 1D. 01 Mask requests when M2D_COUNT reaches zero, 2D. 10 Mask requests when M3D_COUNT reaches zero, 3D. 11 Mask requests when M4D_COUNT reaches zero, 4D.

Table 14-32. BD_MD_2D Field Descriptions

Bits	Description
M2D_COUNT 51–40	**Second Dimension Current Count** Decrements each time the BD_MD_SIZE reaches zero. The field is reloaded with the M2D_BCOUNT each time it reaches zero. **Note:** If the buffer is two dimensional or more, this field cannot be 0.
M2D_BCOUNT 39–28	**Second Dimension Base Count** Holds the second dimension base count. **Note:** If the buffer is more than two dimensional, this field cannot be 0.
M2D_OFFSET 27–0	**Second Dimension Offset** Written in two's complement. The offset is added to the BD_MD_ADDR each time BD_MD_SIZE reaches zero.

Table 14-33. BD_MD_3D Field Descriptions

Bits	Description
M3D_COUNT 51–40	**Third Dimension Current Count** Decrements each time the BD_MD_SIZE and M2D_COUNT reach zero. The field is reloaded with the M3D_BCOUNT each time it reaches zero. **Note:** If the buffer is three dimensional or more, this field cannot be 0.
M3D_BCOUNT 39–28	**Third Dimension Base Count** Holds the third dimension base count. **Note:** If the buffer is more than three dimensional, this field cannot be 0.
M3D_OFFSET 27–0	**Third Dimension Offset** Written in two's complement. The offset is added to the BD_MD_ADDR each time BD_MD_SIZE and M2D_COUNT reach zero.

Table 14-34. BD_MD_4D Field Descriptions

Bits	Description
M4D_COUNT 39–28	**Fourth Dimension Current Count** Decrements each time BD_MD_SIZE, M2D_COUNT, and M3D_COUNT reach zero. **Note:** If the buffer is four dimensional, then this field cannot be 0.
M4D_OFFSET 27–0	**Fourth Dimension Offset** Written in two's complement. The offset is added to the BD_MD_ADDR each time BD_MD_SIZE, M2D_COUNT, and M3D_COUNT reach zero.

High Speed Serial Interface (HSSI) Subsystem

15

The High Speed Serial Interface (HSSI) is a 8-port serial communication subsystem that supports the following multiplexed serial interface combinations:

- Two x1/x4 Serial RapidIO ports
- One x1/x4 Serial RapidIO ports, one x1 Serial RapidIO port, and two SGMII ports
- One x1/x4 Serial RapidIO port and a PCI Express port
- One x1/x4 Serial RapidIO port, two SGMII ports, and a PCI Express port

To support these interfaces, the HSSI includes the following blocks:

- One RapidIO controller with two ports and one RapidIO Messaging Unit (RMU)
- One PCI Express controller with a bridge to the OCN fabric.
- One 8-port OCN fabric with two DMA controllers to connect between the RapidIO and PCI Express controllers and the system CLASS module
- Two OCN-to-MBus (O2M) bridges to link the HSSI to the system CLASS module
- Two SRIO port controllers to link the RMU and OCN fabric to the SerDes ports.
- Two SerDes interfaces to connect to the external signal interface.

These communication interfaces allow the cores to execute the data processing code and be relieved from the data transfer and handling overhead for processing serial data flow.

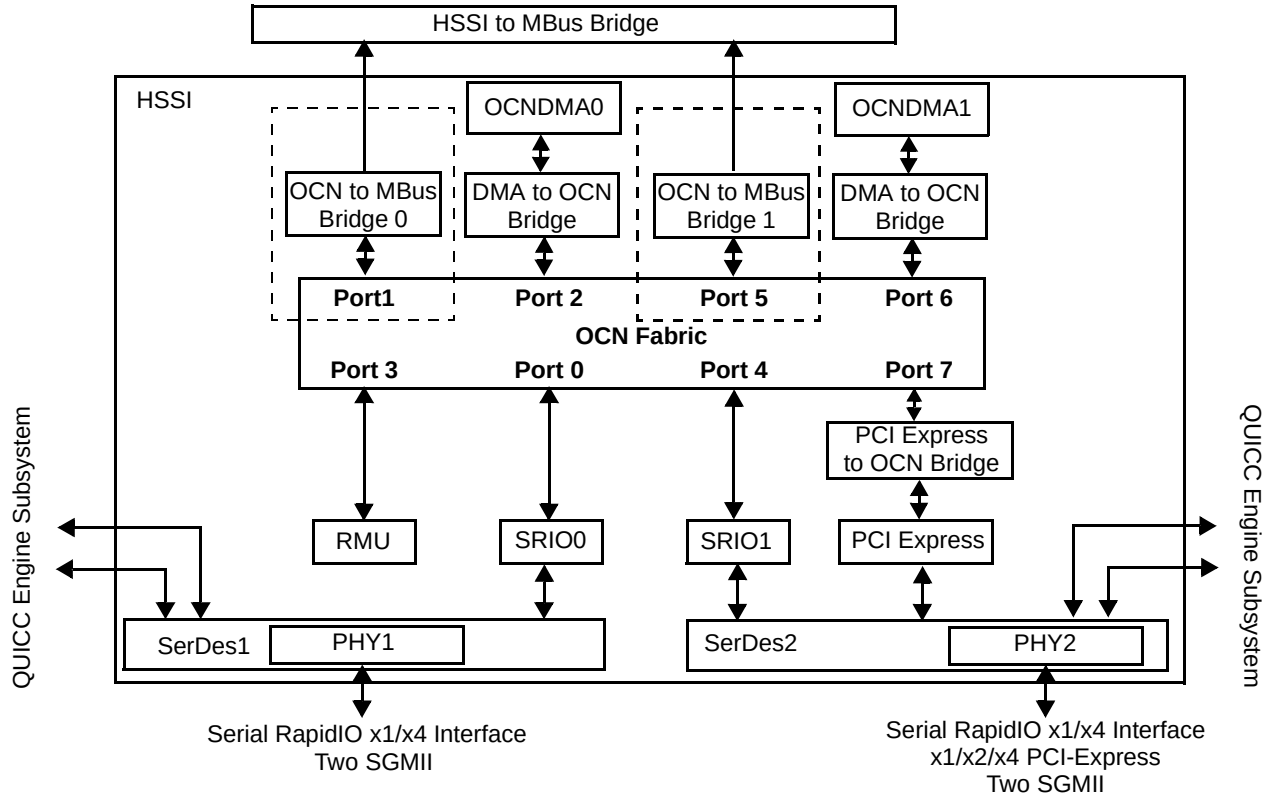
This chapter describes the communication interface components which include the following:

- OCN fabric
- DMA controllers
- OCN-to-MBus bridges
- SRIO interface modules
- SerDes interfaces

This chapter includes the interface component programming model, with the exception of the SerDes multiplexing programming, which is done using the S1P and S2P bits in the low half of the reset configuration word (RCW), as described in **Chapter 5**, *Reset*. The communication controllers supported within the block are described in **Chapter 16**, *Serial RapidIO® Controller* and **Chapter 17**, *PCI Express Controller*. While the SGMII lines are multiplexed through the SerDes interfaces, they are functionally part of the QUICC Engine subsystem, as described in **Chapter 18**, *QUICC Engine Subsystem*.

15.1 HSSI Subsystem Block Diagram

Figure 15-1 shows a block diagram of the HSSI.



Notes: **1.** The actual signals multiplexed for each PHY is determined by the SerDes configuration field contents in the lower 32 bits of the reset configuration word, which are recorded in RCWLR[S1P] and RCWLR[S2P]. See **Chapter 5**, *Reset* for details.

2. You must distribute the access loading between O2M0 (Port 1) and O2M1 (Port 5) for optimal performance. Although the internal OCN arbiters attempt to balance access automatically, these ports can be configured to work specifically with individual RapidIO inbound windows (see **Section 16.6.59** for details) and PCI Express inbound windows (see **Section 17.4.1.4.13** for details).

Figure 15-1. HSSI Block Diagram

15.2 OCN Fabric

The On-Chip Network (OCN) fabric is a non-blocking high speed interconnect used for embedded system devices. The MSC8156E DSP HSSI uses an 8-port OCN to connect between the Serial RapidIO Controller, the PCI Express controller, the two supporting DMA controllers and the dual x4 SerDes PHYs. The OCN requires no programming and provides a seamless interface for the HSSI.

The OCN arbitration for each OCN target port is based on the following two stages of “least recently used” decisions:

- Stage 1 selects the least recently used source from among all different sources with the same priority level. This is done for each of the four possible priority levels, yielding up to four “winners” (one for each priority level).
- Stage 2 selects the least recently used source from the four winners of stage 1.

Using this method prevents starvation of a lower priority source, since older source will win over newer source which have a higher priority.

As an example, given a case of two OCN sources (A and B) that are constantly targeting the same OCN port (C), one source (A) is using only priority 0 and the other one (B) is using only priority 1. Each decision cycle, the priority 0 winner is source A, and the priority 1 winner is source B. The next stage winner alternates between A and B. Therefore, the order of the final winners of the arbitration are A, B, A, B, and so on, regardless of the fact that B has the higher priority.

15.3 DMA Controllers

The MSC8156E includes two dedicated DMA controllers that transfer blocks of data between the serial RapidIO controller and the PCI Express controller and the local address space independent from the DSP cores. **Figure 15-1** shows the block diagram of each dedicated DMA controller.

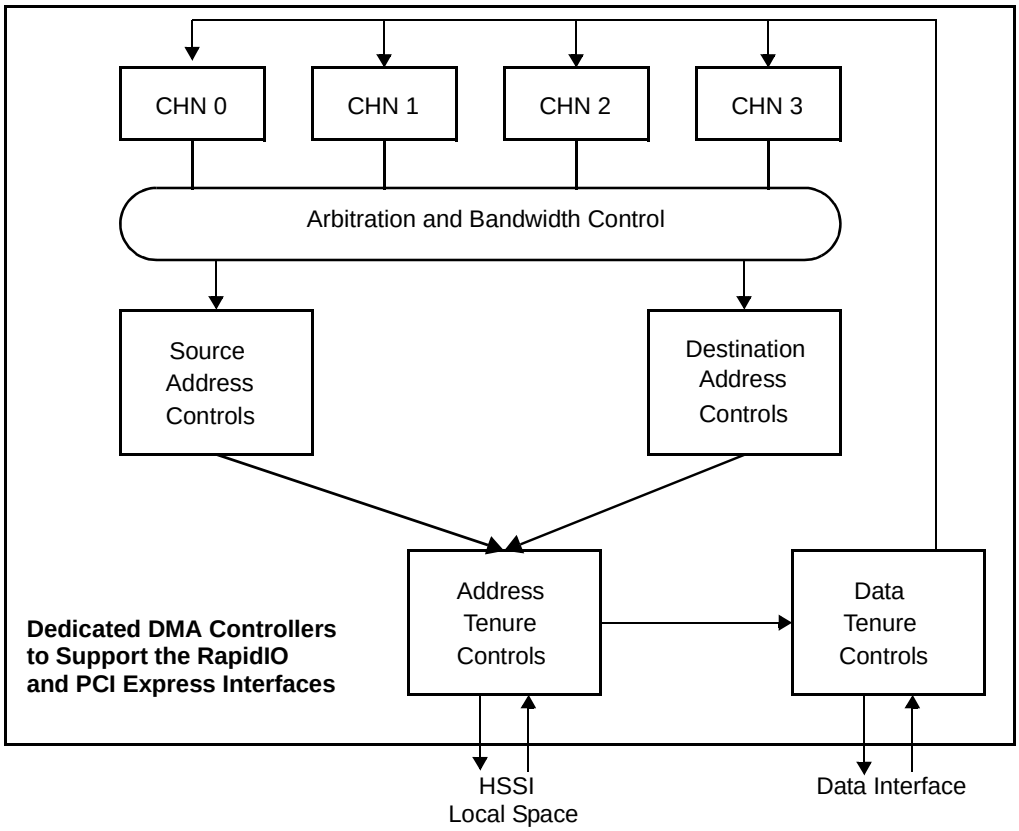


Figure 15-2. Interface Dedicated DMA Controller Block Diagram

15.3.1 Overview

Each dedicated DMA controller has four high-speed DMA channels. Each of the DSP cores can initiate DMA transfers. All 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.

15.3.2 Features

Each dedicated DMA controller offers the following features:

- Four 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
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- An Address Translation Management Unit (ATMU) with 10 local access address windows. The ATMU translates a request address into a logical device source/destination.

15.3.3 Modes of Operation

Each MSC8156E dedicated DMA controller has two modes of operation: basic and extended. Basic mode is the DMA legacy mode. It does not support advanced features. Extended mode supports advanced features like striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. **Table 15-1** 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 15-1** before starting a transfer.
- *Chaining mode*. Software must initialize descriptors in memory and the required fields as described in **Table 15-1** before starting a transfer.

- *Single-write start mode*. The DMA process can be started by 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 15-1. 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)
	BC (basic chaining)	BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
	ExtC (extended chaining)	ExtCS (ExtC single-write start)

Refer to **Section 15.4**, *Functional Description* for details on these modes. **Figure 15-3** shows the general DMA operational flow chart.

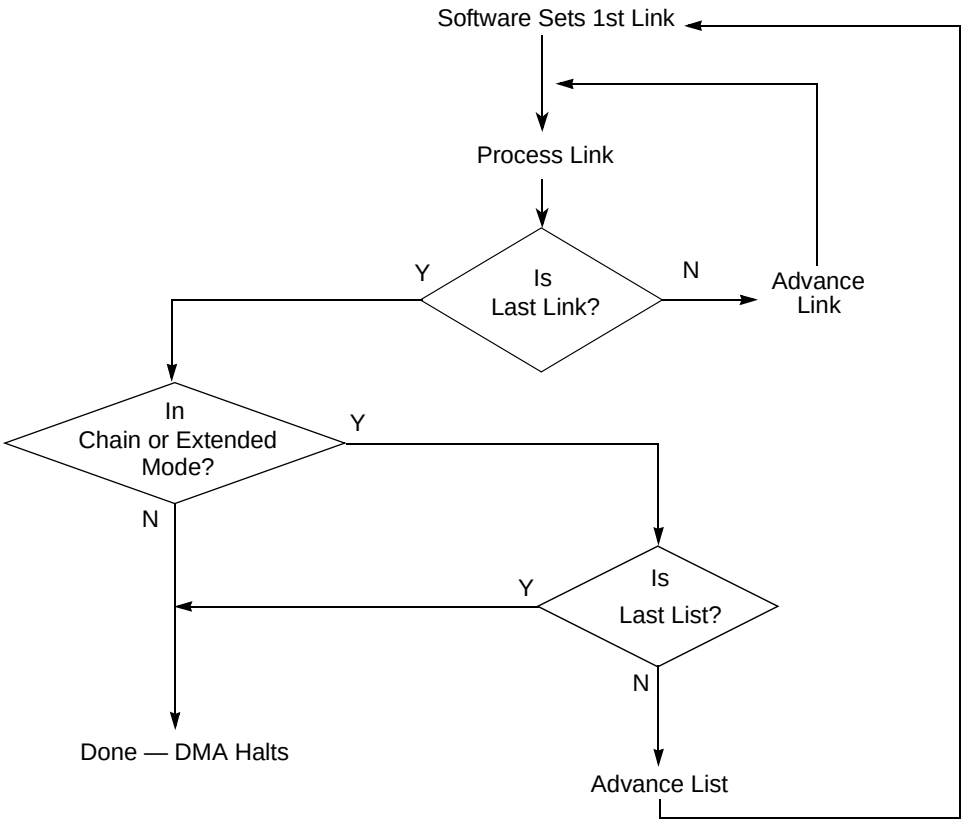


Figure 15-3. DMA Operational Flow Chart

15.4 Functional Description

This section describes the function of the DMA controller.

15.4.1 DMA Channel Operation

All DMA channels support two different modes of operation, a basic mode (MRn[XFE] = 0) and an extended mode (MRn[XFE] = 1). In both modes, a channel can be activated by clearing and setting MRn[CS] or through the single-write start mode using MRn[CDSM/SWSM] and MRn[SRW].

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 **Section 15.4.1.1**. The DMA always reads/writes the maximum number of bytes for a given transfer as described by the capability inputs of the DMA controller. Using 256 bytes over the RapidIO interface reduces packet overhead that 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 controller is designed to support RapidIO transaction types, including various priority level support. The DMA controller offers reads that can be mapped to non-coherent (NREAD) or maintenance reads. In addition, the writes can be mapped to non-coherent (NWRITE, NWRITE_R, SWRITE) writes and maintenance writes. 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 15-2** for more complete descriptions of the channel states and state transitions.

15.4.1.1 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 in **Figure 15-4**

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].

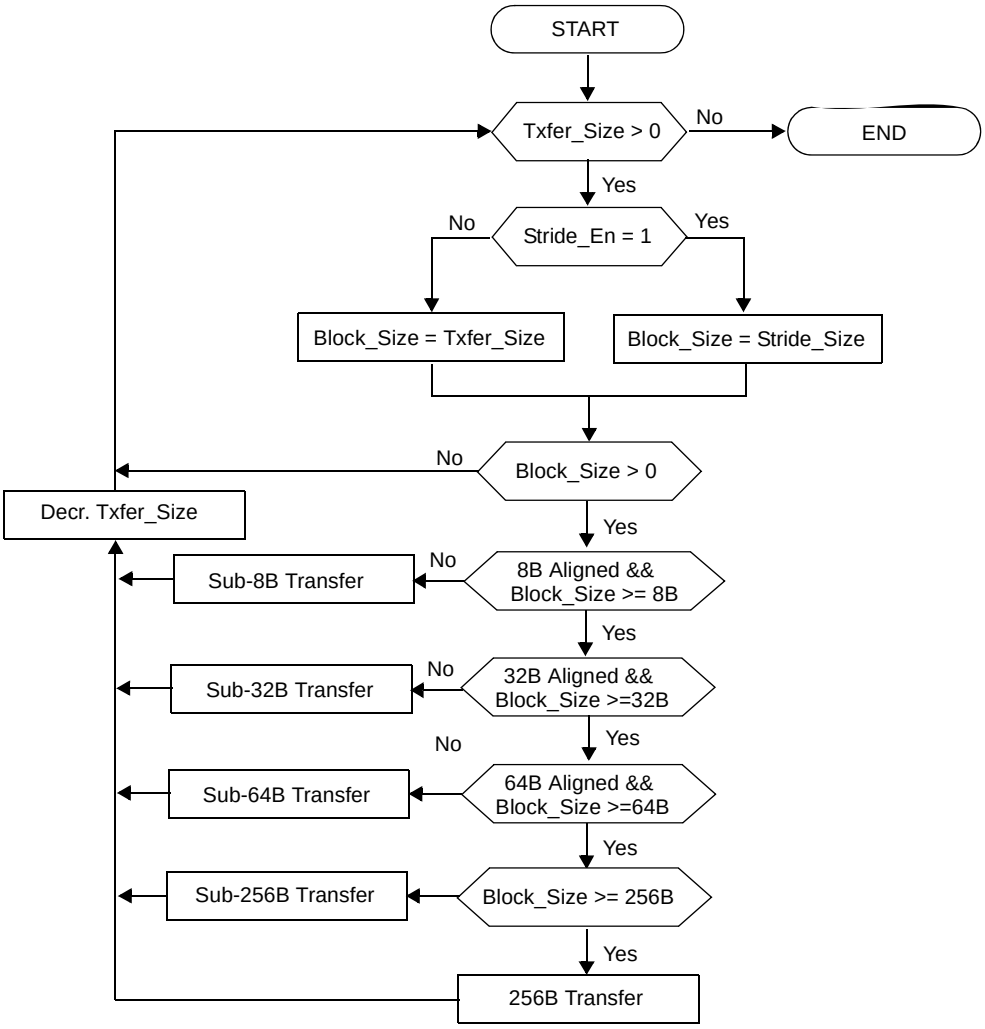


Figure 15-4. Source/Destination Engine Transaction Size Flow Chart

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 this example, 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

15.4.1.2 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.

15.4.1.2.1 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 15-2**), 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.

15.4.1.2.2 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 15-2**), 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, 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.

15.4.1.2.3 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 15-2**), 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, 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, 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.

15.4.1.2.4 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*). 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 MRn[CDSM/SWSM]. Clear the extended features enable bit MRn[XFE] and the channel transfer mode bit, MRn[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 15-2**), 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, 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.

15.4.1.2.5 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].

15.4.1.2.6 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*.

15.4.1.2.7 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*.

15.4.1.2.8 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 15-2**), 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, 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, 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.

15.4.1.2.9 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. 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], MR*n*[CTM], and MR*n*[XFE] 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 15-2**), 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.

15.4.1.3 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 get started 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.

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.

15.4.1.3.1 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.

15.4.1.3.2 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.

15.4.1.4 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.

15.4.1.5 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 that any channel will transfer before re-arbitration is 1 Kbyte.

15.4.1.6 Channel State

Table 15-2 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 15-2. Channel State Table

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[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 MR*n*[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

15.4.1.7 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. **Figure 15-5** illustrates the stride size and distance parameters. 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.

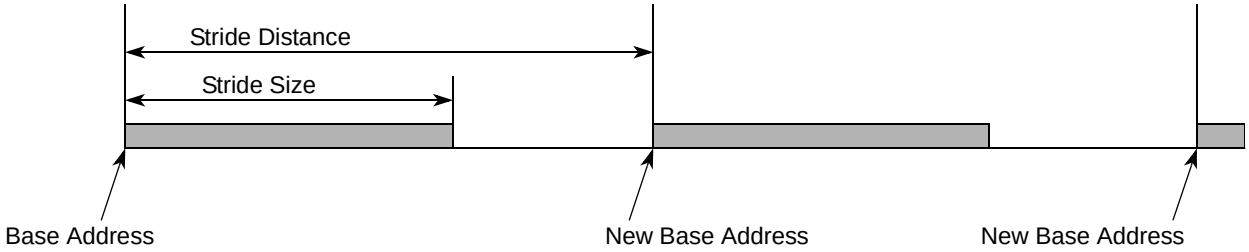


Figure 15-5. Stride Size and Stride Distance

15.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

15.4.3 DMA Errors

On a transfer error (address mapping errors, for example), 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
- Transfer started with a priority of three
- Illegal type, defined by SATR*n*[SREADTTYPE] and DATR*n*[DWRITETTYPE], used for the transfer.

15.4.4 DMA Descriptors

The DMA engine recognizes list descriptors and link 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 EOLND bit in the next link descriptor; 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.

Note: Software must ensure that each descriptor is aligned on a 32-byte boundary.

The last link descriptor in the last list in memory sets NLNDAR*n*[EOLND] 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 as shown in. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor.

Table 15-3 summarizes the DMA list descriptors.

Table 15-3. 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

Table 15-4 summarizes the DMA link descriptors.

Table 15-4. Link DMA Descriptor Summary

Descriptor Field	Description
Source attributes register	Contains source transaction attributes (SATR).
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 (SAR).
Destination attributes register	Contains destination transaction attributes (DATR).
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 (DAR).
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.
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 (BCR).

Figure 15-6 shows the format of the list descriptors for 32-bit devices.

Offset	Field
0x00	Reserved
0x04	Next List Descriptor Address
0x08	Reserved
0x0c	First Link Descriptor Address
0x10	Source Stride
0x14	Destination Stride
0x18	Reserved
0x1c	Reserved

Figure 15-6. List Descriptor Format (Used for 32-bit devices)

Note: For each channel (DMA controller 0 or 1, channels 0–3), the values written to the mode register determine how the DMA controller executes the data transfers, such as a simple direct transfer, a chain of direct transfers specified by a set of links, or a complex chain involving chained lists, each of which contains pointers to chained link sets (as indicated by **Table 15-3** and **Table 15-4**). **Figure 15-6** and **Figure 15-7** show the order in which data should be stored in memory (for link descriptors and link descriptors, respectively); the name listed for each field indicates the register format to use for the data written to the specified offset. For example, *Next List Descriptor Address* refers to the **Next List Descriptor Address Register (DnNLSDARx)** for the specified DMA controller (n = 0 or 1) and the specific channel (x = 0–3). See the specified register description in **Section 15.8**, *HSSI Programming Model* for details.

Figure 15-7 shows the format of the link descriptors for 32-bit devices.

Offset	Field
0x00	Source Attributes
0x04	Source Address
0x08	Destination Attributes
0x0c	Destination Address
0x10	Reserved
0x14	Next Link Descriptor Address
0x18	Byte Count
0x1c	Reserved

Figure 15-7. Link Descriptor Format (Used for 32-bit devices

15.4.5 Local Access ATMU Registers

The local access ATMU has ten address windows that can map a DMA request to a programmable transaction interface (logical device). The user set up the ATMU windows correctly before enabling translation through the ATMU. In a multiple hit scenario, the first matching window is used. If no hit is detected or if mapping is to a reserved transaction interface, the source/target defaults to local address space.

Note: See **Section 15.8.19**, *Local Access Window Base Address Registers 0–9 (DnLAWBAR[0–9])*, on page 15-46 and **Section 15.8.20**, *Local Access Window Attributes Registers 0–9 (DnLAWAR[0–9])*, on page 15-47 for details.

15.4.6 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the dedicated DMA controller and is intended to help software maximize the DMA performance and avoid DMA programming errors.

The DMA controller restrictions are as follows:

- Due to the limited number of buffers that the DMA controller can use, stride sizes less than 64 bytes should be avoided. Maximum utilization is obtained from strides greater than or equal to 256 bytes. However, small stride sizes can be used for scatter-gather functions.
- 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.
- If the DMA is programmed to send SWRITEs over RapidIO, the programmer must ensure that the destination address is double-word aligned and that the byte count is a double-word multiple.
- A single DMA transfer in any of the direct or chaining modes must not cross a 16 GB (34-bit) address boundary.

15.5 OCN-to-MBus (O2M) Bridges

The O2M bridges provide an interface between the OCN fabric and the MBus interfaces that connect internally to the CLASS. The bridges support all mandatory MBus and OCN interface protocol procedures. The bridges provide initiator interfaces to access the target MBus, formats OCN packets, negotiates with the OCN arbiter, and transmits/receives associated transactions. Accesses are configurable as 32-byte or 64-byte transfers using the OCN-to-MBus Control Registers (O2MCR[0–1]) as described in **Section 15.8.21**.

15.6 SRIO Port Controller Modules (SRIOn)

The SRIO0 and SRIO1 modules provide an interface bridge between the RMU, the two serial RapidIO ports, and the two SerDes PHYs. The units require no programming, but transfers through the SRIO modules can be tracked by the performance monitor. See **Section 27.3**, *Performance Monitor* in **Chapter 27**, *Debugging, Profiling, and Performance Monitoring* for details.

15.7 SerDes PHY Interfaces

The HSSI includes two 4-port SerDes PHY interfaces that are multiplexed between the two Serial RapidIO ports, the PCI Express port, and the two SGMII ports from the QUICC Engine subsystem. Multiplexing configuration is done using the S1P and S2P fields in the RCWLR. See **Section 5.3.1** in **Chapter 5**, *Reset* for multiplexing configuration details. Control of the individual lanes is done through the SRDS Control Register 2 described in **Section 15.8.29**.

15.8 HSSI Programming Model

The following sections describe the dedicated DMA controller 0 and 1 registers and the SerDes PHY interface control registers.

The majority of the DMA controller registers are channel-specific and can be identified by one of the four offsets that describe the register. These registers include the following:

- Dn DMA 0–3 Mode Registers (DnMR[0–3]), page 15-24
- Dn DMA 0–3 Status Registers (DnSR[0–3]), page 15-27
- Dn DMA 0–3 Current Link Descriptor Extended Address Registers (DnECLNDAR[0–3], page 15-29
- Dn DMA 0–3 Current Link Descriptor Address Registers (DnCLNDAR[0–3]), page 15-30
- Dn DMA 0–3 Source Attributes Registers (DnSATR[0–3]), page 15-31
- Dn DMA 0–3 Source Address Registers (DnSAR[0–3]), page 15-32
- Dn DMA 0–3 Destination Attributes Registers (DnDATR[0–3]), page 15-33

- Dn DMA 0–3 Destination Address Registers (DnDAR[0–3]), page 15-34
- Dn DMA 0–3 Byte Count Registers (DnBCR[0–3], page 15-35
- Dn DMA 1–3 Next Link Descriptor Extended Address Registers (DnENLNDAR[0–3], page 15-36
- Dn DMA 0–3 Next Link Descriptor Address Registers (DnNLNDAR[0–3]), page 15-37
- Dn DMA 1–3 Current List Descriptor Extended Address Registers (DnECLSDAR[0–3], page 15-38
- Dn DMA 0–3 Current List Descriptor Address Registers (DnCLSDAR[0–3]), page 15-39
- Dn DMA 0–3 Next List Descriptor Extended Address Registers (DnENLSDAR[0–3], page 15-40
- Dn DMA 0–3 Next List Descriptor Address Registers (DnNLSDAR[0–3]), page 15-41
- Dn DMA 0–3 Source Stride Registers (DnSSR[0–3]), page 15-42
- Dn DMA 0–3 Destination Stride Registers (DnDSR[0–3]), page 15-43
- Dn DMA General Status Register (DnDGSR), page 15-44
- Dn Local Access Window Base Address Registers 0–9 (DnLAWBAR[0–9]), page 15-46
- Dn Local Access Window Attributes Registers 0–9 (DnLAWAR[0–9]), page 15-47

Note: The dedicated DMA Controller 0 (D0) registers use a base address of 0xFFFA8000. The dedicated DMA Controller 1 (D1) registers use a base address of 0xFFFAA000.

The OCN-to-MBus Control Registers use a set of registers in each bridge to control the data width of the burst transfers. The registers are:

- OCN-to-MBus Configuration Registers 0–1 (O2MCR[0–1]), page 15-49
- OCN-to-MBus Error Attribute Registers 0–1 (O2MEAR[0–1]), page 15-50
- OCN-to-MBus Error Address Registers 0–1 (O2MEADR[0–1]), page 15-52
- OCN-to-MBus Error Status Registers 0–1 (O2MESR[0–1]), page 15-53
- OCN-to-MBus Interrupt Enable Registers 0–1 (O2MIER[0–1]), page 15-54
- OCN-to-MBus Error Capture Enable Registers 0–1 (O2MECER[0–1]), page 15-55

Note: The O2M bridge 0 uses a base address of 0xFFFA1000. The O2M bridge 1 uses a base address of 0xFFFA1040

The SerDes PHYs use control registers in each PHY to control the individual data lanes. The registers are:

- SRDSn Control Register 0 for Channel n (SRDSnCR0_[1–2]. page 15-56
- SRDSn Control Register 1 for Channel n (SRDSnCR1_[1–2]. page 15-59
- SRDSn Control Register 2 for Channel n (SRDSnCR2_[1–2], page 15-62
- SRDSn Control Register 3 for Channel n (SRDSnCR3_[1–2]. page 15-63
- SRDSn Control Register 4 for Channel n (SRDSnCR4_[1–2].page 15-65

- SRDSn Control Register 5 for Channel n (SRDSnCR5_[1–2]. page 15-67
- SRDSn Control Register 6 for Channel n (SRDSnCR6_[1–2].page 15-68

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

There are three general configuration registers used to control and monitor HSSI operation (see **Chapter 8**, *General Configuration Registers* for details. These are the following:

- High Speed Serial Interface Status Register (HSSI_SR), page 8-7
- High Speed Serial Interface Control Register 1 (HSSI_CR1), page 8-12
- High Speed Serial Interface Control Register 2 (HSSI_CR2), page 8-15

Note: The base address for the general configuration registers is: 0xFFF28000.

15.8.1 Mode Registers 0–3 (DnMR[0–3]).

DnMR0 Mode Registers 0–3 Offset 0x100
DnMR1 Offset 0x180
DnMR2 Offset 0x200
DnMR3 Offset 0x280

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				BWC				—						DAHTS	
Type	R/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
	SAHTS		DAHE	SAHE	—	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM/ SWSM	CA	CTM	CC	CS
	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. **Table 15-5** describes the fields of the DnMR.

Table 15-5. DnMR Field Descriptions

Bits	Reset	Description	Setting
— 31–28	0	Reserved. Write to zero for future compatibility.	
BWC 27–24	0	**Bandwidth Control** If multiple channels are executing transfers concurrently, this value determines how many bytes a channel can transfer before the DMA controller shifts to the next channel. If a single channel is executing, configure this field as 1111 for the channel.	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 Bandwidth sharing disabled; allows uninterrupted transfers from each channel.
— 23–18	0	Reserved. Write to zero for future compatibility.	
DAHTS 17–16	0	**Destination Address Hold Transfer Size** Indicates the transfer size to use while MR[DAHE} is set. The byte count must be in multiples of the size and the destination address must be aligned based on the size. The defined size must be equal to or small than the value of MR[BWC] to avoid undefined behavior.	00 1 byte. 01 2 bytes. 10 4 bytes. 11 8 bytes.

Table 15-5. DnMR Field Descriptions (Continued)

Bits	Reset	Description	Setting
SATHS 15–14	0	**Source Address Hold Transfer Size** Indicates the transfer size to use while MR[SAHE} is set. The byte count must be in multiples of the size and the destination address must be aligned based on the size. The defined size must be equal to or small than the value of MR[BWC] to avoid undefined behavior.	00 1 byte. 01 2 bytes. 10 4 bytes. 11 8 bytes.
DAHE 13	0	**Destination Address Hold Enable** When set, allows the DMA controller to hold the destination address of a transfer to the size specified by DATHS. This hardware feature only supports aligned transfers.	0 Destination address hold disabled. 1 Destination address hold enabled.
SAHE 12	0	**Source Address Hold Enable** When set, allows the DMA controller to hold the destination address of a transfer to the size specified by SATHS. This hardware feature only supports aligned transfers.	0 Source address hold disabled. 1 Source address hold enabled.
— 11	0	Reserved. Write to zero for future compatibility.	
SRW 10	0	**Single Register Write (CTM = 1 only)** The effect of setting this bit in direct mode depends on the value of CDSM/SWSM. **Note:** This bit is reserved for CTM = 0 (chaining mode).	*CDSM/SWSM = 0* 0 Normal operation. 1 A write to the destination address register sets MR[CS] to initiate a DMA transfer. *CDSM/SWSM = 1* 0 Normal operation. 1 A write to the source address register sets MR[CS] to initiate a DMA transfer.
EOSIE 9	0	**End-of-Segments interrupt Enable** When set, generates an interrupt to indicate the completion of a data transfer. **Note:** When set, the value of this bit overrides the value of CLNDAR[EOSIE] on a link descriptor basis.	0 No end-of-transfer interrupt generated. 1 End-of-transfer interrupt generated.
EOLNIE 8	0	**End-of-Links Interrupt Enable** When set, generates an interrupt at the completion of a list of DMA transfers (that is, sets NLNDAR[EOLND]).	0 No end-of-list interrupt generated. 1 End-of-list interrupt generated.
EOLSIE 7	0	**End-of-Lists Interrupt Enable** When set, generates an interrupt at the completion of all DMA transfers (that is, sets NLNDAR[EOLND] and NLSDAR[EOLSD]).	0 No end of all transfers interrupt generated. 1 End of all transfers interrupt generated.
EIE 6	0	**Error Interrupt Enable** When set, generates an interrupt if a programming or transfer error is detected.	0 No error interrupt generated. 1 Error interrupt generated.
XFE 5	0	**Extended Chaining Enable (CTM = 0 only)** When set, enables extended chaining mode. **Note:** This bit is reserved in direct mode.	0 Extended chaining disabled. 1 Extended chaining enabled.

Table 15-5. DnMR Field Descriptions (Continued)

Bits	Reset	Description	Setting
CDSM/SWSM 4	0	**Current Descriptor Start Mode/Single-Write Start Mode** The function of this bit varies depending on the setting of XFE, CTM, and SRW. **Note:** This bit must be cleared when SRW is cleared.	CTM = 0 and XFE = 0 0 Normal operation. 1 Single-write start mode in which a write to the current link descriptor address register sets MR[CS] to start the DMA transfer. CTM = 0 and XFE = 1 0 Normal operation. 1 Single-write start mode in which a write to the current list descriptor address register sets MR[CS] to start the DMA transfer. *CTM = 1 and SRW = 0* 0 Normal operation. 1 A write to the current link descriptor address register sets MR[CS] to initiate a DMA transfer. *CTM = 1 and SRW = 1* 0 A write to the destination address register sets MR[CS] to initiate a DMA transfer. 1 A write to the source address register sets MR[CS] to initiate a DMA transfer.
CA 3	0	**Channel Abort** When set, causes the channel to abort the transfer and clear CB. The channel then remains idle until a new transfer is programmed.	0 No effect. 1 Abort current transfer.
CTM 2	0	**Channel Transfer Mode** When set, configures the controller in direct mode, which means that software must place all the required parameters into the necessary registers to start the DMA transfer.	0 Chaining mode. 1 Direct mode.
CC 1	0	**Channel Continue (chaining mode only)** When set, restarts the transferring process starting at the current descriptor address. This bit is reserved in external master mode. The bit is cleared automatically by hardware after the first descriptor read when continuing a transfer.	0 No effect. 1 Restarts the DMA transfer at the current descriptor address.
CS 0	0	**Channel Start** Stops or starts the DMA transfer. This bit is set automatically by hardware during single-write start mode and external master start enable mode.	0 Stops the DMA transfer if the channel is busy (SR[CB] is set), no effect if the channel is idle. 1 Starts the DMA process if the channel is idle (SR[CB] is cleared). Setting the bit while the channel is busy continues the current transfer from the point at which it stopped.

15.8.2 Status Registers (DnSR*n*)

DnSR0 Status Registers 0–3 Offset 0x104
DnSR1 Offset 0x184
DnSR2 Offset 0x204
DnSR3 Offset 0x284

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—								TE	—	CH	PE	EOLNI	CB	EOSI	EOLSI
	R/W								W1C	R/W	R	W1C	W1C	R	W1C	W1C
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The status registers report various DMA conditions during and after a DMA transfer. **Table 15-6** describes the fields of the DnSR.

Table 15-6. DnSR Field Descriptions

Bits	Reset	Description	Setting
— 31–8	0	Reserved. Write to zero for future compatibility.	
TE 7	0	**Transfer Error** Indicates whether an error occurred during the DMA transfer. See **Section 15.4.3**, *DMA Errors,* on page 15-16 for details. **Note:** Write a 1 to this bit to clear it.	0 No error during the DMA transfer. 1 Error condition during the DMA transfer.
— 6	0	Reserved. Write to zero for future compatibility.	
CH 5	0	**Channel Halted** Indicates whether the transfer is halted. Attempts to halt a channel that is idle have no effect. If the bit is set, the channel was successfully halted by software and can be restarted.	0 Channel is not halted. 1 DMA successfully halted by software.
PE 4	0	**Programming Error** Indicates whether a programming error was detected that prevented the DMA transfer. **Note:** Write a 1 to this bit to clear it.	0 No programming error detected. 1 Programming error detected.
EOLNI 3	0	**End-of-Links Interrupt** After transferring the last block of data in the last link descriptor. if MR[EOLSIE] is set, then this bit is set and an interrupt is generated. **Note:** Write a 1 to this bit to clear it.	0 No end-of-links interrupt. 1 End-of-links interrupt.
CB 2	0	**Channel Busy** Indicates the current status of the channel.	0 Channel is idle, DMA transfer completed, error occurred, or a channel abort occurred. 1 DMA transfer is in progress.

Table 15-6. DnSR Field Descriptions (Continued)

Bits	Reset	Description	Setting
EOSI 1	0	**End-of-Segment Interrupt** In chaining mode, after finishing a data transfer, if MR[EOLSIE] is set or if CLNDAR[EOSIE} is set, then this bit is set and an interrupt is generated. In direct mode, if MR[EOSIE] is set, then this bit is set and an interrupt is generated. **Note:** Write a 1 to this bit to clear it.	0 No end-of-segment interrupt. 1 End-of-segment interrupt.
EOLSI 0	0	**End-of-List Interrupt** After transferring the last block of data in the last list descriptor, if MR[EOLSIE] is set, then this bit is set and an interrupt is generated. **Note:** Write a 1 to this bit to clear it.	0 No end-of-list interrupt. 1 End-of-list interrupt.

15.8.3 Current Link Descriptor Extended Address Registers (DnECLNDAR*n*)

DnECLNDAR0 Current Link Descriptor Extended Address Registers 0–3 Offset 0x108
DnECLNDAR1 Offset 0x188
DnECLNDAR2 Offset 0x208
DnECLNDAR3 Offset 0x288

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

The DnECLNDAR contains the extended address of the current link descriptor for the specified channel.

Note: These registers are only used for RapidIO transactions. They are not used for accesses to the internal RapidIO address space.

For RapidIO transactions in basic chaining mode, software must initialize this register and the Current Link Descriptor Address Register (DnCLNDAR) to point to the first link descriptor in memory. After the current descriptor is processed, the DnECLNDAR and DnCLNDAR are loaded from the Next Link Descriptor Extended Address Register (DnENLNDAR) and the Next Link Descriptor Address Register (DnNLNDAR). Then the controller evaluates the DnNLNDAR*n*[EOLND] field. If EOLND is cleared (0), the DMA controller reads in the new current link descriptor for processing. If EOLND is set (1), the last descriptor of the list has 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 the EOLSD bit in the next list descriptor address register (DnNLSDAR). If EOLSD is clear, the controller loads the contents of the DnENLSDAR into the Current List Descriptor Extended Address Register (DnECLSDAR) and the contents of the DnNLSDAR into the DnCLSDAR and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts. **Table 15-7** describes the DnECLNDAR fields.

Table 15-7. DnECLNDAR Field Descriptions

Bits	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	

Table 15-7. DnECLNDAR Field Descriptions (Continued)

Bits	Reset	Description	Setting
ECLNDA 3–0	0	**Current Link Descriptor Extended Address** Contains the most significant 4 bits of the 36-bit address used with RapidIO transactions only. **Note:** This field is not used for local transactions.	

15.8.4 Current Link Descriptor Address Registers (DnCLNDAR*n*):

DnCLNDAR0 Current Link Descriptor Address Registers 0–3 Offset 0x10C
DnCLNDAR1 Offset 0x18C
DnCLNDAR2 Offset 0x20C
DnCLNDAR3 Offset 0x28C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLNDA															
Type	R/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**
	CLNDA												EOSIE	—		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DnCLNDAR contains the address of the current link descriptor for the specified channel. In basic chaining mode, software must initialize this register to point to the first link descriptors in memory. After the current descriptor is processed, the CLDAR is loaded from the NLNDAR and the NLNDAR*n*[EOLND] field in the DnNLNDAR 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 DnNLSDAR*n*[EOLSD] in the DnNLSDAR. If EOLSD is clear, the controller loads the contents of the DnNLSDAR into the DnCLNDAR and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts. **Table 15-8** describes the DnCLNDAR fields.

Table 15-8. DnCLNDAR Field Descriptions

Bits	Reset	Description	Setting
CLNDA 31–4	0	**Current Link Descriptor Address** Holds the current descriptor address of the buffer descriptor in memory. The descriptor must be 32-byte aligned. **Note:** This field is used for all transfers. For RapidIO transactions, it is the lower portion of the 36-bit address formed by combining with the ECLNDA for use with RapidIO transaction types.	

Table 15-8. DnCLNDAR Field Descriptions (Continued)

Bits	Reset	Description	Setting
EOSIE 3	0	**End-of-Segment Interrupt Enable** Enables/disables the end-of-segment interrupt at the completion of the current DMA transfer for the current link descriptor.	0 Do not generate end-of-segment interrupt. 1 Generate end-of-segment interrupt when transfer is complete.
— 2–0	0	Reserved. Write to zero for future compatibility.	

15.8.5 Source Attributes Registers (DnSATR*n*)

DnSATR0 Source Attributes Registers 0–3 Offset 0x110
DnSATR1 Offset 0x190
DnSATR2 Offset 0x210
DnSATR3 Offset 0x290

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—							SSME	—				SREADTTYPE			
Type	R/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
	—												ESAD			
	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DnATR contains the transaction attributes to be used for the DMA operation on the specified channel. Stride mode is enabled by setting DnSATR[SSME]. Source read transaction type is specified using DnSATR[SREADTTYPE]. Attributes are derived from local access windows and outbound ATMU registers according to the transaction address.

Table 15-9 describes the fields of the DnSATR.

Table 15-9. DnSATR Field Descriptions

Bits	Reset	Description	Setting
— 31–25	0	Reserved. Write to zero for future compatibility.	
SSME 24	0	**Source Stride Mode Enable** Enables/disables source stride mode. When enabled, you must set the required stride size and distance in the Source Stride Register (SSR) for the specified channel. **Note:** This bit is ignored in basic mode (MR[EFE] is cleared (0).	0 Stride mode disabled. 1 Stride mode enabled.
— 23–20	0	Reserved. Write to zero for future compatibility.	
SREADTTYPE 19–16	0	**DMA Source Transaction Type** Specifies the source transaction type. **Note:** Writing a reserved value to this field causes a programming error to be detected and indicated in SR[PE] for the specified channel.	0100 Read. All other values reserved.

Table 15-9. DnSATR Field Descriptions (Continued)

Bits	Reset	Description	Setting
— 15–4	0	Reserved. Write to zero for future compatibility.	
ESAD 3–0	0	**Extended Source Address** Represents the most significant 4 bits of the 36-bit source address of the DMA transfer with SARx.	

15.8.6 Source Address Registers (DnSAR*n*)

DnSAR0 Source Address Registers 0–3 Offset 0x114
DnSAR1 Offset 0x194
DnSAR2 Offset 0x214
DnSAR3 Offset 0x294

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

The SAR contains the address from which the DMA controller reads data for the specified channel. In direct mode, if DnMR[CDSM/SWSM] and DnMR[SRW] are set, a write to this register simultaneously sets DnMR[CS], starting a DMA transfer. Software must ensure that this is a valid address. **Table 15-10** describes the DnSAR fields.

Table 15-10. DnSAR Field Descriptions

Bits	Reset	Description
Local Source		
SAD 31–0	0	**Source Address** Contains least significant 32 bits of the 36-bit source 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, it which case it remains equal to the address from with the last stride began.

15.8.7 Destination Attributes Registers (DnDATR*n*).

DnDATR0 Destination Attributes Registers 0–3 Offset 0x118
DnDATR1 Offset 0x198
DnDATR2 Offset 0x218
DnDATR3 Offset 0x298

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—	NLWR	—					DSME	—				DWRITETTYPE			
Type	R/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
	—												EDAD			
	R/W															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The destination attributes registers contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DnDATR[DSME] for the specified channel. Destination write transaction type is specified using the DnDATR[DWRITETTYPE] field.

The target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DnDATR[DWRITETTYPE] using the local address space category.

Table 15-11 describes the fields of the DATR.

Table 15-11. DnDATR Field Descriptions

Bits	Reset	Description	Setting
— 31	0	Reserved. Write to zero for future compatibility.	
NLWR 30	0	**No Last Write With Response** Used to indicate whether the last write requires a target response. The ROWARx[WRTYP] value selects the write transaction type. If NLWR is cleared, then the last write transaction is NWRITE_R instead of SWRITE or NWRITE, depending on the value of ROWARx[WRTYP]. If NLWR is set, the last write transaction is performed as specified by the value of ROWARx[WRTYP].	0 Last write transfer is a write with target response. 1 Last write transfer is a write without target response.
— 20–25	0	Reserved. Write to zero for future compatibility.	
DSME 24	0	**Destination Stride Mode Enable** Enables/disables destination stride mode. When enabled, you must set the required stride size and distance in the Destination Stride Register (DSR) for the specified channel. **Note:** This bit is ignored in basic mode (MR[EFE] is cleared (0).	0 Stride mode disabled. 1 Stride mode enabled.
— 23–20	0	Reserved. Write to zero for future compatibility.	

Table 15-11. DnDATR Field Descriptions (Continued)

Bits	Reset	Description	Setting
DWRITETTYPE 19–16	0	**DMA Destination Transaction Type** Specifies the destination transaction type. **Note:** Writing a reserved value to this field causes a programming error to be detected and indicated in SR[PE] for the specified channel.	0100 Write. All other values reserved.
— 15–4	0	Reserved. Write to zero for future compatibility.	
EDAD 3–0	0	**Extended Destination Address** Represents the most significant 4 bits of the 36-bit destination address of the DMA transfer with DARx.	

15.8.8 Destination Address Registers (DnDAR*n*)

DnDAR0 Destination Address Registers 0–3 Offset 0x11C
DnDAR1 Offset 0x19C
DnDAR2 Offset 0x21C
DnDAR3 Offset 0x29C

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

The DnDAR contains the address to which the DMA controller writes data for the specified channel. In direct mode, if DnMR[CDSM/SWSM] is cleared and DnMR[SRW] is set, a write to this register simultaneously sets DnMR[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Table 15-12 describes the DnDAR fields.

Table 15-12. DnDAR Field Descriptions

Bits	Reset	Description
Local Source		
DAD 31–0	0	**Destination Address** Contains the least significant 32 bits of the 36-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, it which case it remains equal to the address from with the last stride began.

15.8.9 Byte Count Registers (DnBCR*n*).

DnBCR0 Byte Count Registers 0–3 Offset 0x120
DnBCR1 Offset 0x1A0
DnBCR2 Offset 0x220
DnBCR3 Offset 0x2A0

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

The byte count register contains the number of bytes to transfer. **Table 15-13** describes the fields of the DnBCR.

Table 15-13. DnBCR Field Descriptions

Bits	Reset	Description
— 31–26	0	Reserved. Write to zero for future compatibility.
BC 25–0	0	**Byte Count** Contains the number of bytes to transfer. The value in this field is decremented after each DMA read operation. The maximum transfer size is 2^{26} – 1 bytes.

15.8.10 Extended Next Link Descriptor Address Registers (DnENLNDAR*n*)

DnENLNDAR1 Extended Next Link Descriptor Address Registers 1–3 Offset 0x1A4
DnENLNDAR2 Offset 0x224
DnENLNDAR3 Offset 0x2A4

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

The DnENLNDAR contains the extended address of the next link descriptor for the specified channel.

Note: These registers are only used for RapidIO transactions. They are not used for accesses to the internal RapidIO address space.

For RapidIO transactions in basic chaining mode, software must initialize this register and the NLNDAR to point to the next link descriptor in memory. After the current descriptor is processed, the ECLNDAR and CLNDAR are loaded from the ENLNDAR and the NLNDAR. Then the controller evaluates the NLNDAR*n*[EOLND] field. If EOLND is cleared (0), the DMA controller reads in the new current link descriptor for processing. If EOLND is set (1), the last descriptor of the list has 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 the EOLSD bit in the next list descriptor address register (NLSDAR). If EOLSD is clear, the controller loads the contents of the ENLSDAR into the ECLSDAR and the contents of the NLSDAR into the CLSDAR and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts. **Table 15-14** describes the ENLNDAR fields.

Table 15-14. DnENLNDAR Field Descriptions

Bits	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	
ENLNDA 3–0	0	**Next Link Descriptor Extended Address** Contains the most significant 4 bits of the 36-bit address used with RapidIO transactions only. **Note:** This field is not used for local transactions.	

15.8.11 Next Link Descriptor Address Registers (DnNLNDAR*n*)

DnNLNDAR0 Next Link Descriptor Address Registers 0–3 Offset 0x128
DnNLNDAR1 Offset 0x1A8
DnNLNDAR2 Offset 0x228
DnNLNDAR3 Offset 0x2A8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NLNDA															
Type	R/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**
	NLNDA											—	NDEOSIE	—		EOLND
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DnNLNDAR contains the address of the next link descriptor for the specified channel. In basic chaining mode, software must initialize this register to point to the first link descriptors in memory. After the current descriptor is processed, the CLDAR is loaded from the DnNLNDAR and the DnNLNDAR*n*[EOLND] field in the DnNLNDAR 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 DnNLSDAR*n*[EOLSD] in the DnNLSDAR. If EOLSD is clear, the controller loads the contents of the NLSDAR into the CLNDAR and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts. **Table 15-15** describes the DnNLNDAR fields.

Table 15-15. DnNLNDAR Field Descriptions

Bits	Reset	Description	Setting
NLNDA 31–5	0	**Next Link Descriptor Address** Holds the descriptor address of the next buffer descriptor in memory. The descriptor must be 32-byte aligned. **Note:** This field is used for all transfers. For RapidIO transactions, it is the lower portion of the 36-bit address formed by combining with the ENLNDA for use with RapidIO transaction types.	
— 4	0	Reserved. Write to zero for future compatibility.	
NDEOSIE 3	0	**Next Descriptor End-of-Segment Interrupt Enable** Enables/disables the next descriptor end-of-segment interrupt when the current DMA transfer for the current link descriptor completes.	0 Do not generate next descriptor end-of-segment interrupt. 1 Generate next descriptor end-of-segment interrupt when transfer is complete.
— 2–1	0	Reserved. Write to zero for future compatibility.	

Table 15-15. DnNLNDAR Field Descriptions (Continued)

Bits	Reset	Description	Setting
EOLND 0	0	**End-of-Links Descriptor** Indicates whether the descriptor is the last descriptor in memory for this list. **Note:** This bit is ignored in direct mode.	0 Not the last descriptor for this list. 1 Last descriptor for this list.

15.8.12 Extended Current List Descriptor Address Registers (DnECLSDAR*n*)

DnECLSDAR1 Extended Current List Descriptor Address Registers 0–3 Offset 0x1B0
DnECLSDAR2 Offset 0x230
DnECLSDAR3 Offset 0x2B0

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

The DnECLSDAR contains the extended address of the current address of the list descriptor in memory in extended chaining mode for the specified channel.

Note: These registers are only used for RapidIO transaction types. They are not used for accesses to the internal RapidIO address space.

In extended chaining mode, software must initialize this register and DnCLSDAR 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 DnENLNDAR and the DnNLNDAR. Then the controller evaluates the DnNLSDAR*n*[EOLSD] field. If EOLSD is cleared (0), the DMA controller reads in the new current list descriptor for processing. If EOLND is set (1) and the last link of the current list is finished, all DMA transfers are complete. **Table 15-16** describes the DnECLSDAR fields.

Table 15-16. DnECLSDAR Field Descriptions

Bits	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	
ECLSDA 3–0	0	**Current List Descriptor Extended Address** Contains the most significant 4 bits of the 36-bit address used with RapidIO transactions only. **Note:** This field is not used for local transactions.	

15.8.13 Current List Descriptor Address Registers (DnCLSDAR*n*)

DnCLSDAR0 Current List Descriptor Address Registers 0–3 Offset 0x134
DnCLSDAR1 Offset 0x1B4
DnCLSDAR2 Offset 0x234
DnCLSDAR3 Offset 0x2B4

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

The DnCLSDAR contains the address of the current list descriptor for the specified channel. In extended chaining mode, software must initialize this register 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 DnENLSDAR and the DnNLSDAR. Then the controller evaluates the DnNLSDAR*n*[EOLSD] field. If EOLSD is cleared (0), the DMA controller reads in the new current list descriptor for processing. If EOLND is set (1) and the last link of the current list is finished, all DMA transfers are complete. **Table 15-17** describes the DnCLSDAR fields.

Table 15-17. DnCLSDAR Field Descriptions

Bits	Reset	Description	Setting
CLSDA 31–5	0	**Current List Descriptor Address** Holds the current list descriptor address of the buffer descriptor in memory. The descriptor must be 32-byte aligned. **Note:** This field is used for all transfers. For RapidIO transactions, it is the lower portion of the 36-bit address formed by combining with the ECLSDA for use with RapidIO transaction types.	
— 4–0	0	Reserved. Write to zero for future compatibility.	

15.8.14 Extended Next List Descriptor Address Registers (DnENLSDAR*n*)

DnENLSDAR0 Extended Next List Descriptor Address Registers 0–3 Offset 0x138
DnENLSDAR1 Offset 0x1B8
DnENLSDAR2 Offset 0x238
DnENLSDAR3 Offset 0x2B8

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

The DnENLSDAR contains the extended address of the next address of the 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 for the specified channel.

Note: These registers are only used for RapidIO transaction types. They are not used for accesses to the internal RapidIO address space.

Table 15-18 describes the DnENLSDAR fields.

Table 15-18. DnENLSDAR Field Descriptions

Bits	Reset	Description	Setting
— 31–4	0	Reserved. Write to zero for future compatibility.	
ENLSDA 3–0	0	**Next List Descriptor Extended Address** Contains the most significant 4 bits of the 36-bit address used with RapidIO transactions only. **Note:** This field is not used for local transactions.	

15.8.15 Next List Descriptor Address Registers (DnNLSDAR*n*)

DnNLSDAR0 Next List Descriptor Address Registers 0–3 Offset 0x13C
DnNLSDAR1 Offset 0x1BC
DnNLSDAR2 Offset 0x23C
DnNLSDAR3 Offset 0x2BC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NLSDA															
Type	R/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**
	NLSDA											—				EOLSD
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DnNLSDAR contains the address of the next address of the 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 for the specified channel. **Table 15-19** describes the DnNLSDAR fields.

Table 15-19. DnNLSDAR Field Descriptions

Bits	Reset	Description	Setting
NLSDA 31–5	0	**Next List Descriptor Address** Holds the next list descriptor address of the buffer descriptor in memory. The descriptor must be 32-byte aligned. **Note:** This field is used for all transfers. For RapidIO transactions, it is the lower portion of the 36-bit address formed by combining with the ENLSDA for use with RapidIO transaction types.	
— 4–1	0	Reserved. Write to zero for future compatibility.	
EOLSD 0	0	**End-of-Lists Descriptor** Indicates whether the descriptor is the last list descriptor in memory. When the bit is set, the DMA controller halts after the last link descriptor transaction finishes. **Note:** This bit is ignored in direct mode.	0 Not the last list descriptor in memory. 1 Last list descriptor in memory.

15.8.16 Source Stride Registers (DnSSR*n*)

DnSSR0 Source Stride Registers 0–3 Offset 0x140
DnSSR1 Offset 0x1C0
DnSSR2 Offset 0x240
DnSSR3 Offset 0x2C0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								SSS							
Type	R/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**
	SSS				SSD											
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DnSSR contains the source stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the DnSSR applies for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated. **Table 15-19** describes the DnSSR fields.

Table 15-20. DnSSR Field Descriptions

Bits	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
SSS 23–12	0	**Source Stride Size** Holds the number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
SSD 11–0	0	**Source Stride Distance** The source stride distance in bytes from the start byte to the end byte.

15.8.17 Destination Stride Registers (DnDSR*n*)

DnDSR0 Destination Stride Registers 0–3 Offset 0x144
DnDSR1 Offset 0x1C4
DnDSR2 Offset 0x244
DnDSR3 Offset 0x2C4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								DSS							
Type	R/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**
	DSS				DSD											
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DnDSR contains the destination stride size and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the DnDSR applies for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated. **Table 15-21** describes the DnDSR fields.

Table 15-21. DnDSR Field Descriptions

Bits	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
DSS 23–12	0	**Destination Stride Size** Holds the number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
DSD 11–0	0	**Destination Stride Distance** The destination stride distance in bytes from the start byte to the end byte.

15.8.18 DMA General Status Register (DnDGSR))

DnDGSR DMA General Status Register Offset 0x300

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TE0	—	CH0	PE0	EOLNI0	CB0	EOSI0	EOLSI0	TE1	—	CH1	PE1	EOLNI1	CB1	EOSI1	EOLSI1
Type	R															
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**
	TE2	—	CH2	PE2	EOLNI2	CB2	EOSI2	EOLSI2	TE3	—	CH3	PE3	EOLNI3	CB3	EOSI3	EOLSI3
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DMA general status register combines all of the status bits from each channel into one register. This register is read-only. **Table 15-22** describes the fields of the DnDGSR.

Table 15-22. DnDGSR Field Descriptions

Bits	Reset	Description	Setting
TE0 31	0	**Channel 0 Transfer Error** Indicates whether a transfer error occurred.	0 Normal operation. 1 Error condition occurred during DMA transfer.
— 30	0	Reserved. Write to zero for future compatibility.	
CH0 29	0	**Channel 0 Halted** Indicates whether the channel halted.	0 Channel not halted. 1 Channel halted.
PE0 28	0	**Channel 0 Programming Error Detected** Indicates whether a programming error was detected.	0 Normal operation. 1 Programming error detected.
EOLNI0 27	0	**Channel 0 End-of-Links Interrupt** Indicates whether an end-of-links interrupt occurred.	0 Normal operation. 1 End-of-links interrupt occurred.
CB0 26	0	**Channel 0 Busy** Indicates whether the channel is busy.	0 Channel not busy. 1 Channel busy.
EOSI0 25	0	**Channel 0 End-of-Segment Interrupt** Indicates whether an end-of-segment interrupt occurred.	0 Normal operation. 1 End-of-segment interrupt occurred.
EOLSI0 24	0	**Channel 0 End-of-Lists/Direct Interrupt** Indicates whether an end-of lists/direct interrupt occurred.	0 Normal operation. 1 End-of-list/direct interrupt occurred.
TE1 23	0	**Channel 1 Transfer Error** Indicates whether a transfer error occurred.	0 Normal operation. 1 Error condition occurred during DMA transfer.
— 22	0	Reserved. Write to zero for future compatibility.	
CH1 21	0	**Channel 1 Halted** Indicates whether the channel halted.	0 Channel not halted. 1 Channel halted.
PE1 20	0	**Channel 1 Programming Error Detected** Indicates whether a programming error was detected.	0 Normal operation. 1 Programming error detected.
EOLNI1 19	0	**Channel 1 End-of-Links Interrupt** Indicates whether an end-of-links interrupt occurred.	0 Normal operation. 1 End-of-links interrupt occurred.

Table 15-22. DnDGSR Field Descriptions (Continued)

Bits	Reset	Description	Setting
CB1 18	0	**Channel 1 Busy** Indicates whether the channel is busy.	0 Channel not busy. 1 Channel busy.
EOSI1 17	0	**Channel 1 End-of-Segment Interrupt** Indicates whether an end-of-segment interrupt occurred.	0 Normal operation. 1 End-of-segment interrupt occurred.
EOLSI1 16	0	**Channel 1 End-of-Lists/Direct Interrupt** Indicates whether an end-of lists/direct interrupt occurred.	0 Normal operation. 1 End-of-list/direct interrupt occurred.
TE2 15	0	**Channel 2 Transfer Error** Indicates whether a transfer error occurred.	0 Normal operation. 1 Error condition occurred during DMA transfer.
— 14	0	Reserved. Write to zero for future compatibility.	
CH2 13	0	**Channel 2 Halted** Indicates whether the channel halted.	0 Channel not halted. 1 Channel halted.
PE2 12	0	**Channel 2 Programming Error Detected** Indicates whether a programming error was detected.	0 Normal operation. 1 Programming error detected.
EOLNI2 11	0	**Channel 2 End-of-Links Interrupt** Indicates whether an end-of-links interrupt occurred.	0 Normal operation. 1 End-of-links interrupt occurred.
CB2 10	0	**Channel 2 Busy** Indicates whether the channel is busy.	0 Channel not busy. 1 Channel busy.
EOSI2 9	0	**Channel 2 End-of-Segment Interrupt** Indicates whether an end-of-segment interrupt occurred.	0 Normal operation. 1 End-of-segment interrupt occurred.
EOLSI2 8	0	**Channel 2 End-of-Lists/Direct Interrupt** Indicates whether an end-of lists/direct interrupt occurred.	0 Normal operation. 1 End-of-list/direct interrupt occurred.
TE3 7	0	**Channel 3 Transfer Error** Indicates whether a transfer error occurred.	0 Normal operation. 1 Error condition occurred during DMA transfer.
— 6	0	Reserved. Write to zero for future compatibility.	
CH3 5	0	**Channel 3 Halted** Indicates whether the channel halted.	0 Channel not halted. 1 Channel halted.
PE3 4	0	**Channel 3 Programming Error Detected** Indicates whether a programming error was detected.	0 Normal operation. 1 Programming error detected.
EOLNI3 3	0	**Channel 3 End-of-Links Interrupt** Indicates whether an end-of-links interrupt occurred.	0 Normal operation. 1 End-of-links interrupt occurred.
CB3 2	0	**Channel 3 Busy** Indicates whether the channel is busy.	0 Channel not busy. 1 Channel busy.
EOSI3 1	0	**Channel 3 End-of-Segment Interrupt** Indicates whether an end-of-segment interrupt occurred.	0 Normal operation. 1 End-of-segment interrupt occurred.
EOLSI3 0	0	**Channel 3 End-of-Lists/Direct Interrupt** Indicates whether an end-of lists/direct interrupt occurred.	0 Normal operation. 1 End-of-list/direct interrupt occurred.

15.8.19 Local Access Window Base Address Registers 0–9 (DnLAWBAR[0–9])

DnLAWBAR0 Local Access Window Base Address Registers 0–9 Offset 0x1C08
DnLAWBAR1 Offset 0x1C28
DnLAWBAR2 Offset 0x1C48
DnLAWBAR3 Offset 0x1C68
DnLAWBAR4 Offset 0x1C88
DnLAWBAR5 Offset 0x1CA8
DnLAWBAR6 Offset 0x1CC8
DnLAWBAR7 Offset 0x1CE8
DnLAWBAR8 Offset 0x1D08
DnLAWBAR9 Offset 0x1D28

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

DnLAWBARx holds 24 most significant bits of the window base address. The least significant byte is always 0s. **Table 15-23** describes the DnLAWBARx fields.

Table 15-23. DnLAWBARx Field Descriptions

Bits	Reset	Description	Setting
— 31–24	0	Reserved. Write to zero for future compatibility.	
BA 23–0	0	**Base Address** Holds the 24 most significant bits of the 36-bit window base address. • Bits 23–20 correspond to the value of SATRx[ESAD]/DATRx[EDAD] • Bits 19–0 correspond to bits 31–12 of SARx[SAD]/DARx[DAD] **Note:** For local transactions, the most significant 4 bits in this field must be 0s.	

15.8.20 Local Access Window Attributes Registers 0–9 (DnLAWAR[0–9])

DnLAWAR0	Local Access Window Attributes Registers 0–9	Offset 0x1C10
DnLAWAR1		Offset 0x1C30
DnLAWAR2		Offset 0x1C50
DnLAWAR3		Offset 0x1C70
DnLAWAR4		Offset 0x1C90
DnLAWAR5		Offset 0x1CB0
DnLAWAR6		Offset 0x1CD0
DnLAWAR7		Offset 0x1CF0
DnLAWAR8		Offset 0x1D10
DnLAWAR9		Offset 0x1D30

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	—							TRANS_INT				—			
Type	R/W	R							R/W				R			
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
	—										SIZE					
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DnLAWARx contains the transaction interface for a request mapped to this window. **Table 15-24** describes the DnLAWARx fields.

Table 15-24. DnLAWARx Field Descriptions

Bits	Reset	Description	Settings
EN 31	0	**Enable** Enables/disables the local access window (LAW) and all other LAWAR and LAWBAR fields. When the LAW is enabled, the LAWAR and LAWBAR fields combine to define the address range for this window.	0 Local access window disabled. 1 Local access window enabled.
— 30–24	0	Reserved. Write to zero for future compatibility.	
TRANS_INT 23–20	0	**Transaction Interface** Specifies the logical destination/target of the transaction mapped to this window. A reserved value defaults to local address space.	0001 PCI Express 1011 OCN to MBus Bridge 0 for local space 1100 SRIO Port 0 1101 SRIO Port 1 1111 OCN to MBus Bridge 1 for local space. All others reserved.
— 19–6	0	Reserved. Write to zero for future compatibility.	

Table 15-24. DnLAWARx Field Descriptions (Continued)

Bits	Reset	Description	Settings	
SIZE 5–0	0	**Local Access Window Size** This value determines the local access window, which is computed using the formula $2^{(SIZE + 1)}$. The minimum size is 4K. The maximum size is 64G.	001011 001100 001101 001110 001111 010000 010001 010010 010011 010100 010101 010110 010111 011000 011001 011010 011011 011100 011101 011110 011111 100000 100001 100010 100011 All others reserved.	4K 8K 16K 32K 64K 128K 256K 512K 1M 2M 4M 8M 16M 32M 64M 128M 256M 512M 1G 2G 4G 8G 16G 32G 64G

15.8.21 OCN-to-MBus Configuration Registers (O2MCR[0–1])

O2MCR0 OCN-to-MBus Configuration Registers 0–1 Offset 0x00
O2MCR1 Offset 0x00

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

O2MCRn is used to control the size of the burst transactions between the OCN and the MBus.

Note: The O2M bridge 0 uses a base address of 0xFFFA1000.
The O2M bridge 1 uses a base address of 0xFFFA1040.

Table 15-33 describes the O2MCRn fields.

Table 15-25. O2MCRn Field Descriptions

Bits	Reset	Description	Settings
— 31–19	0b0000000100000	Reserved. All bits in this field are set by default and must not be changed.	
MBS 18–16	101	**Maximum Burst Size** Determines the maximum burst width for MBus transactions.	101 64 bytes. 110 128 bytes. 111 256 bytes All other values reserved.
— 15–0	0x0	Reserved. Write to zero for future compatibility.	

15.8.22 OCN-to-MBus Error Attribute Registers (O2MEAR[0–1])

O2MEAR0 OCN-to-MBus Error Attribute Registers 0–1 Offset 0x10
O2MEAR1 Offset 0x10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ERR_ADDR				ERR_ID				—						WR	RWB
Type	R															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—			REQ_TYPE					—		ERR_SRC					
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

O2MEARn locks the attributes of an incorrect transaction, such as the ID of the source that initiates the transaction and the request type. The capture of a specific error attribute is performed only if the corresponding it in the Error Capture Enable Register is set. Once the capture occurs, the register is locked. You can release the register to capture another error by clearing the corresponding status bit in the O2MESR through a register write.

Note: The O2M bridge 0 uses a base address of 0xFFFA1000.
The O2M bridge 1 uses a base address of 0xFFFA1040.

Table 15-33 describes the O2MEARn fields.

Table 15-26. O2MEARn Field Descriptions

Bits	Reset	Description	Settings
ERR_ADDR 30–28	0	**Error Address** Four most significant bits of the captured address. These are used with the 32 least significant bits captured O2MEADRn to derive the full 36-bit address of the erroneous transaction.	
ERR_ID 27–24	0	**Error Identification** This field is used to identify the type of error captured.	0000 No error attributes captured. 0001 DE, transfer error, OCN arbiter timeout. 0010 TE, Early EOP error. 0011 EE, Transaction Type error. 0100 TT, OCN data error. 0101 MD, MBus data error. 0110 ME, MBus EOT error All other values reserved.
— 23–18	0	Reserved. Always write zeros to these bits for future compatibility.	
WR 17	0	**With Response** Indicates whether the OCN request packet is with or without response.	0 Without response. 1 With response.
RWB 16	0	**Read/Write** Indicates the type of access (read/write).	0 Write access executed. 1 Read access executed.
— 15–13	0	Reserved. Write to zero for future compatibility.	

Table 15-26. O2MEARn Field Descriptions (Continued)

Bits	Reset	Description	Settings
REQ_TYPE 12–8	0	**Transaction Request Type** Indicates the request type of the incorrect transaction.	00000 Write with no response (if WR=0) or write with response (if WR=1). 00001 Read with response (WR=1) 00010 Configuration write with response (WR=1) 00011 Configuring read with response (WR=1) 00100 Message with response (WR=1) 00101 Doorbell with response (WR=1) 00110 User defined transaction with no response (WR=0) or User-defined transaction with response (WR=1) 01100 Read and post increment with response (WR=1) 01101 Read and post decrement with response (WR=1) 01110 Read and set to all 1s with response (WR=1) 01111 Read and clear to all 0x with response (WR=1) All other values reserved.
— 7–6	0x0	Reserved. Write to zero for future compatibility.	
ERR_SRC 5–0	0	**Error Source ID** The unique ID of the OCN initiator that performs the error-causing transaction.	

15.8.23 OCN-to-MBus Error Address Registers (O2MEADR[0–1])

O2MEADR0 OCN-to-MBus Error Address Registers 0–1 Offset 0x14
O2MEADR1 Offset 0x14

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

O2MEADRn indicates the least significant bits of the captured address of the incorrect transaction. These 32 bits are combined with the 4 most significant bits in the O2MEARn to yield the 36-bit address. The register contents are locked once the address is captured and can only be cleared by writing to the corresponding status bit in the O2MESRn.

Note: The O2M bridge 0 uses a base address of 0xFFFA1000.
The O2M bridge 1 uses a base address of 0xFFFA1040.

Table 15-33 describes the O2MEADRn fields.

Table 15-27. O2MEADRn Field Descriptions

Bits	Reset	Description	Settings
ERR_ADDR 31–0	0	**Erroneous Transaction Address** Contains the least significant 32 bits of the captured transaction address.	

15.8.24 OCN-to-MBus Error Status Registers (O2MESR[0–1])

O2MESR0 OCN-to-MBus Error Status Registers 0–1 Offset 0x18
O2MESR1 Offset 0x18

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												DE	TE	EE	TT
Type	R												W1C			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															ME
Type	R															W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

O2MESRn stores the error status bits. The bits are set whenever the appropriate error conditions are met. They are cleared by writing a 1 to a set bit. Writing a zero has no effect. Each error status bit has a corresponding interrupt enable bit in the O2MIERn. If the bit is enabled, the hardware generates an interrupt.

Note: The O2M bridge 0 uses a base address of 0xFFFA1000.
The O2M bridge 1 uses a base address of 0xFFFA1040.

Table 15-33 describes the O2MESRn fields.

Table 15-28. O2MESRn Field Descriptions

Bits	Reset	Description	Settings
— 31–20	0	Reserved. Always write zeros to these bits for future compatibility.	
DE 19	0	**Data Error** Indicates whether the error bit is set in the OCN packet data payload.	0 No error detected. 1 Error detected.
TE 18	0	**Transfer Error Acknowledge** Indicates whether an OCN transfer error acknowledge signal was asserted on the inbound interface.	0 No error detected. 1 Error detected.
EE 17	0	**Early End-of-Packet (EOP) Error** Indicates whether the OCN end-of-packet was issued before the expected transaction end.	0 No error detected. 1 Error detected.
TT 16	0	**Transaction Type Error** Indicates whether a transaction type error occurred.	0 No error detected. 1 Error detected.
— 15–1	0	Reserved. Write to zero for future compatibility.	
ME 0	0	**MBus End-of-Transaction Error** Indicates whether an MBus end-of-transaction error occurred.	0 No error detected. 1 Error detected.

15.8.25 OCN-to-MBus Interrupt Enable Registers (O2MIER[0–1])

O2MIER0 OCN-to-MBus Interrupt Enable Registers 0–1 Offset 0x1C
O2MIER1 Offset 0x1C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												DEIE	TEIE	EEIE	TTIE
Type	R												R/W			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															MEIE
Type	R															R/Q
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

O2MIERn enables/disables the interrupts associated with the error status bits. The register is a mask register for the interrupts identified in O2MESRn. Each error status bit has a corresponding interrupt enable bit in the O2MIERn. If the bit is enabled, the hardware generates an interrupt.

Note: The O2M bridge 0 uses a base address of 0xFFFA1000.
The O2M bridge 1 uses a base address of 0xFFFA1040.

Table 15-33 describes the O2MIERn fields.

Table 15-29. O2MIERn Field Descriptions

Bits	Reset	Description	Settings
— 31–20	0	Reserved. Always write zeros to these bits for future compatibility.	
DEIE 19	0	**Data Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled. 1 Interrupt enabled.
TEIE 18	0	**Transfer Error Acknowledge Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled. 1 Interrupt enabled.
EEIE 17	0	**Early End-of-Packet (EOP) Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled. 1 Interrupt enabled.
TTIE 16	0	**Transaction Type Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled. 1 Interrupt enabled.
— 15–1	0	Reserved. Write to zero for future compatibility.	
MEIE 0	0	**MBus End-of-Transaction Error Interrupt Enable** Enables/disables the interrupt.	0 Interrupt disabled. 1 Interrupt enabled.

15.8.26 OCN-to-MBus Error Capture Enable Registers (O2MECER[0–1])

O2MECER0 OCN-to-MBus Error Capture Enable Registers 0–1 Offset 0x20
O2MECER1 Offset 0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												DECE	TECE	EECE	TTCE
Type	R												R/W			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															MECE
Type	R															R/Q
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

O2MECERn enables/disables capturing the address associated with a detected error. Each error status bit has a corresponding capture enable bit in the O2MECERn. If the bit is enabled, the hardware performs an address capture for the error.

Note: The O2M bridge 0 uses a base address of 0xFFFA1000.
The O2M bridge 1 uses a base address of 0xFFFA1040.

Table 15-33 describes the O2MECERn fields.

Table 15-30. O2MECERn Field Descriptions

Bits	Reset	Description	Settings
— 31–20	0	Reserved. Always write zeros to these bits for future compatibility.	
DECE 19	0	**Data Error Capture Enable** Enables/disables address capture.	0 Capture disabled. 1 Capture enabled.
TECE 18	0	**Transfer Error Acknowledge Capture Enable** Enables/disables address capture.	0 Capture disabled. 1 Capture enabled.
EECE 17	0	**Early End-of-Packet (EOP) Error Capture Enable** Enables/disables address capture.	0 Capture disabled. 1 Capture enabled.
TTCE 16	0	**Transaction Type Error Capture Enable** Enables/disables address capture.	0 Capture disabled. 1 Capture enabled.
— 15–1	0	Reserved. Write to zero for future compatibility.	
MECE 0	0	**MBus End-of-Transaction Error Capture Enable** Enables/disables address capture.	0 Capture disabled. 1 Capture enabled.

15.8.27 SRDS Control Register 0 (SRDSnCR0)

SRDSnCR0 SRDS Control Register 0 Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TLCCA	—	RXEQA		TLCCB	—	RXEQB		—							
Type	R/W															
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—	TXEQA			—	TXEQB			—		IACCA	IACCB	—		RXEIA	RXEIB
Type	R/W															
Reset	0	x	x	x	0	x	x	x	0	0	1	1	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR0 contains the functional control bits for the SerDes control logic.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-31 describes the SRDSnCR0 fields.

Note: Lane A corresponds to SerDes lane 0 and Lane B corresponds to SerDes lane 1.

Table 15-31. SRDSnCR0 Field Descriptions

Bits	Reset	Description	
TLCCA 31	0	**Lane A Tracking Loop Centering Control** When enabled, the logic recenters the first stage digital filter after the second stage filter moves the transition point. Recommended setting per protocol: • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Enable recentering algorithm. 1 Disable recentering algorithm.
— 30	0	Reserved. Write to zero for future compatibility.	
RXEQA 29–28	01	**Lane A Receiver Equalization Selection Bus** For PCI Express mode, the value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 01. • SGMII: 01. • Serial RapidIO: 01.	00 No equalization. 01 2 dB of equalization. 10 4 dB of equalization. 11 Reserved.

Table 15-31. SRDSnCR0 Field Descriptions (Continued)

Bits	Reset	Description	
TLCCB 27	0	**Lane B Tracking Loop Centering Control** When enabled, the logic recenters the first stage digital filter after the second stage filter moves the transition point. Recommended setting per protocol: • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Enable recentering algorithm. 1 Disable recentering algorithm.
— 26	0	Reserved. Write to zero for future compatibility.	
RXEQB 25–24	01	**Lane B Receiver Equalization Selection Bus** For PCI Express mode, the value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 01. • SGMII: 01. • Serial RapidIO: 01.	00 No equalization. 01 2 dB of equalization. 10 4 dB of equalization. 11 Reserved.
— 23–15	0	Reserved. Write to zero for future compatibility.	
TXEQA 14–12	xxx	**Lane A Transmitter Equalization Selection Bus** The value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 100. • SGMII: 100. • Serial RapidIO: 011.	000 No equalization. 001 1.09x relative amplitude. 010 1.2x relative amplitude. 011 1.33x relative amplitude. 100 1.5x relative amplitude. 101 1.7x1 relative amplitude. 110 2.0x relative amplitude. 111 Reserved.
— 11	0	Reserved. Write to zero for future compatibility.	
TXEQB 10–8	xxx	**Lane B Transmitter Equalization Selection Bus** The value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 100. • SGMII: 100. • Serial RapidIO: 011.	000 No equalization. 001 1.09x relative amplitude. 010 1.2x relative amplitude. 011 1.33x relative amplitude. 100 1.5x relative amplitude. 101 1.7x1 relative amplitude. 110 2.0x relative amplitude. 111 Reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
IACCA 5	1	**Lane A On-Chip AC Coupling for Receiver** Selects on-chip receiver AC coupling for Lane A. Recommended setting per protocol: • PCI Express: 1. • SGMII: 1. • Serial RapidIO: 1.	0 Disable on-chip AC coupling. 1 Enable on-chip AC coupling.
IACCB 4	1	**Lane B On-Chip AC Coupling for Receiver** Selects on-chip receiver AC coupling for Lane B. Recommended setting per protocol: • PCI Express: 1. • SGMII: 1. • Serial RapidIO: 1.	0 Disable on-chip AC coupling. 1 Enable on-chip AC coupling.
— 3–2	0	Reserved. Write to zero for future compatibility.	

Table 15-31. SRDSnCR0 Field Descriptions (Continued)

Bits	Reset	Description	
RXEIA 1	0	**Lane A Receiver Electrical Idle State** Setting this bit forces the lane into an electrical idle state. • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Lane not forced into an electrical idle state. 1 Lane forced into an electrical idle state.
RXEIB 0	0	**Lane B Receiver Electrical Idle State** Setting this bit forces the lane into an electrical idle state. • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Lane not forced into an electrical idle state. 1 Lane forced into an electrical idle state.

15.8.28 SRDS Control Register 1 (SRDSnCR1)

SRDSnCR1 SRDS Control Register 1 Offset 0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TLCCE	—	RXEQE		TLCCF	—	RXEQF		—							
Type	R/W															
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—	TXEQE			—	TXEQF			—		IACCE	IACCF	—		RXEIE	RXEIF
Type	R/W															
Reset	0	x	x	x	0	x	x	x	0	0	1	1	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR1 contains the functional control bits for the SerDes control logic.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-32 describes the SRDSnCR1 fields.

Note: Lane E corresponds to SerDes lane 2 and Lane F corresponds to SerDes lane 3.

Table 15-32. SRDSnCR1 Field Descriptions

Bits	Reset	Description	
TLCCE 31	0	**Lane E Tracking Loop Centering Control** When enabled, the logic recenters the first stage digital filter after the second stage filter moves the transition point. Recommended setting per protocol: • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Enable recentering algorithm. 1 Disable recentering algorithm.
— 30	0	Reserved. Write to zero for future compatibility.	
RXEQE 29–28	01	**Lane E Receiver Equalization Selection Bus** For PCI Express mode, the value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 01. • SGMII: 01. • Serial RapidIO: 01.	00 No equalization. 01 2 dB of equalization. 10 4 dB of equalization. 11 Reserved.

Table 15-32. SRDSnCR1 Field Descriptions (Continued)

Bits	Reset	Description	
TLCCF 27	0	**Lane F Tracking Loop Centering Control** When enabled, the logic recenters the first stage digital filter after the second stage filter moves the transition point. Recommended setting per protocol: • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Enable recentering algorithm. 1 Disable recentering algorithm.
— 26	0	Reserved. Write to zero for future compatibility.	
RXEQF 25–24	01	**Lane F Receiver Equalization Selection Bus** For PCI Express mode, the value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 01. • SGMII: 01. • Serial RapidIO: 01.	00 No equalization. 01 2 dB of equalization. 10 4 dB of equalization. 11 Reserved.
— 23–15	0	Reserved. Write to zero for future compatibility.	
TXEQE 14–12	xxx	**Lane E Transmitter Equalization Selection Bus** The value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 100. • SGMII: 100. • Serial RapidIO: 011.	000 No equalization. 001 1.09x relative amplitude. 010 1.2x relative amplitude. 011 1.33x relative amplitude. 100 1.5x relative amplitude. 101 1.7x1 relative amplitude. 110 2.0x relative amplitude. 111 Reserved.
— 11	0	Reserved. Write to zero for future compatibility.	
TXEQF 10–8	xxx	**Lane F Transmitter Equalization Selection Bus** The value of this field selects the equalization. Recommended setting per protocol: • PCI Express: 100. • SGMII: 100. • Serial RapidIO: 011.	000 No equalization. 001 1.09x relative amplitude. 010 1.2x relative amplitude. 011 1.33x relative amplitude. 100 1.5x relative amplitude. 101 1.7x1 relative amplitude. 110 2.0x relative amplitude. 111 Reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
IACCE 5	1	**Lane E On-Chip AC Coupling for Receiver** Selects on-chip receiver AC coupling for Lane E. Recommended setting per protocol: • PCI Express: 1. • SGMII: 1. • Serial RapidIO: 1.	0 Disable on-chip AC coupling. 1 Enable on-chip AC coupling.
IACCF 4	1	**Lane F On-Chip AC Coupling for Receiver** Selects on-chip receiver AC coupling for Lane F. Recommended setting per protocol: • PCI Express: 1. • SGMII: 1. • Serial RapidIO: 1.	0 Disable on-chip AC coupling. 1 Enable on-chip AC coupling.
— 3–2	0	Reserved. Write to zero for future compatibility.	

Table 15-32. SRDSnCR1 Field Descriptions (Continued)

Bits	Reset	Description	
RXEIE 1	0	**Lane E Receiver Electrical Idle State** Setting this bit forces the lane into an electrical idle state. • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Lane not forced into an electrical idle state. 1 Lane forced into an electrical idle state.
RXEIF 0	0	**Lane F Receiver Electrical Idle State** Setting this bit forces the lane into an electrical idle state. • PCI Express: 0. • SGMII: 0. • Serial RapidIO: 0.	0 Lane not forced into an electrical idle state. 1 Lane forced into an electrical idle state.

15.8.29 SRDS Control Register 2 (SRDSnCR2)

SRDSnCR2 SRDS Control Register 2 Offset 0x08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—										X3SA	X3SB	—		X3SE	X3SF
Type	R/W															
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—					LBSEL				PLLBW	—					
Type	R/W															
Reset	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR2 is used to enable the signal lanes and select the operating mode of the SerDes interface lanes.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-33 describes the SRDSnCR2 fields.

Note: Lane A corresponds to SerDes lane 0, Lane B corresponds to SerDes lane 1, Lane E corresponds to SerDes lane 2, and Lane F corresponds to SerDes lane 3.

Table 15-33. SRDSnCR2 Field Descriptions

Bits	Reset	Description	
— 31–22	0000000010	Reserved. Write to 0b0000000010 for future compatibility.	
X3SA 21	0	**Lane A Transmitter Tri-State** When set, this bit disables the Lane A transmitter and puts the output into a tri-state (high impedance) condition. Lane A corresponds to channel 0 of the SerDes port.	0 Normal. 1 Lane A transmitter output is disabled and lane is tri-stated.
X3SB 20	0	**Lane B Transmitter Tri-State** When set, this bit disables the Lane B transmitter and puts the output into a tri-state (high impedance) condition. Lane B corresponds to channel 1 of the SerDes port.	0 Normal. 1 Lane B transmitter output is disabled and lane is tri-stated.
— 19–18	0	Reserved. Write to zero for future compatibility.	
X3SE 17	0	**Lane E Transmitter Tri-State** When set, this bit disables the Lane E transmitter and puts the output into a tri-state (high impedance) condition. Lane E corresponds to channel 2 of the SerDes port.	0 Normal. 1 Lane E transmitter output is disabled and lane is tri-stated.
X3SF 16	0	**Lane F Transmitter Tri-State** When set, this bit disables the Lane F transmitter and puts the output into a tri-state (high impedance) condition. Lane F corresponds to channel 3 of the SerDes port.	0 Normal. 1 Lane F transmitter output is disabled and lane is tri-stated.

Table 15-33. SRDSnCR2 Field Descriptions (Continued)

Bits	Reset	Description	
— 15–11	00100	Reserved. Write to 0b00100 for future compatibility.	
LBSEL 10–7	0	**Selects Loop-Back Type** This field selects the loop-back mode for the SerDes port.	0000 Normal operation, no loopback. 0001 Digital loopback mode in both lanes. All other values are reserved.
PLLBW 6	1	**SerDes PLL Bandwidth** Selects the PLL bandwidth.	0 ~4 MHz bandwidth (SGMII and Serial RapidIO protocols). 1 ~8 MHz bandwidth (PCI Express).
— 5–0	0	Reserved. Write to zero for future compatibility.	

15.8.30 SRDS Control Register 3 (SRDSnCR3)

SRDSnCR3 SRDS Control Register 3 Offset 0x0C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/W															
Reset	0	0	0	0	x	x	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**
	—			EICA					—			EICB				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR3 contains functional control bits used by the SerDes logic.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-34 describes the SRDSnCR3 fields.

Note: Lane A corresponds to SerDes lane 0 and Lane B corresponds to SerDes lane 1.

Table 15-34. SRDSnCR3 Field Descriptions

Bits	Reset	Description
— 31–13	0000xx00 00000000 000	Reserved. Write to zero for future compatibility.

Table 15-34. SRDSnCR3 Field Descriptions (Continued)

Bits	Reset	Description	
EICA 12–8	0	**Lane A Receiver Idle Detection Control** The field is broken into two subfields. The highest order 3 bits set the general detection level. The lowest order 2 bits control exit from idle for the PCI Express interface. Recommended setting per protocol: • PCI Express: 10000. • SGMII: 00100. • Serial RapidIO: 00000.	*EICA[12–10]:* 000 Loss of signal detect function disabled. 001 Default SGMII levels (Low = 30 mV, High = 100 mV) 010 Intermediate level (Low =38 mV, High = 120 mV) 011 Intermediate level (Low = 50 mV, High 150 mV) 100 Default PCI Express levels (Low = 65 mV, High = 175 mV) 101 Low = 75 mV, High = 200 mV 110 Intermediate level (Low = 88 mV, High = 225 mV) 111 Intermediate level (Low = 100 mV, High = 250 mV) *EICA[9–8]:* 00 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs (application mode). 01 Exit from Idle ~88 UI and Unexpected Idle Detect ~10 µs 10 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs 11 Bypass
— 7–5	0	Reserved. Write to zero for future compatibility.	
EICB 4–0	0	**Lane B Receiver Idle Detection Control** The field is broken into two subfields. The highest order 3 bits set the general detection level. The lowest order 2 bits control exit from idle for the PCI Express interface. Recommended setting per protocol: • PCI Express: 10000. • SGMII: 00100. • Serial RapidIO: 00000.	*EICB[4–2]:* 000 Loss of signal detect function disabled. 001 Default SGMII levels (Low = 30 mV, High = 100 mV) 010 Intermediate level (Low =38 mV, High = 120 mV) 011 Intermediate level (Low = 50 mV, High 150 mV) 100 Default PCI Express levels (Low = 65 mV, High = 175 mV) 101 Low = 75 mV, High = 200 mV 110 Intermediate level (Low = 88 mV, High = 225 mV) 111 Intermediate level (Low = 100 mV, High = 250 mV) *EICA[1–0]:* 00 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs (application mode). 01 Exit from Idle ~88 UI and Unexpected Idle Detect ~10 µs 10 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs 11 Bypass

15.8.31 SRDS Control Register 4 (SRDSnCR4)

SRDSnCR4 SRDS Control Register 4 Offset 0x10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/W															
Reset	0	0	0	0	x	x	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**
	—			EICE					—			EICF				
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR4 contains functional control bits used by the SerDes logic.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-35 describes the SRDSnCR4 fields.

Note: Lane E corresponds to SerDes lane 2 and Lane F corresponds to SerDes lane 3.

Table 15-35. SRDSnCR4 Field Descriptions

Bits	Reset	Description
— 31–13	0000xx00 00000000 000	Reserved. Write to zero for future compatibility.

Table 15-35. SRDSnCR4 Field Descriptions (Continued)

Bits	Reset	Description	
EICE 12–8	0	**Lane E Receiver Idle Detection Control** The field is broken into two subfields. The highest order 3 bits set the general detection level. The lowest order 2 bits control exit from idle for the PCI Express interface. Recommended setting per protocol: • PCI Express: 10000. • SGMII: 00100. • Serial RapidIO: 00000.	*EICE[12–10]:* 000 Loss of signal detect function disabled. 001 Default SGMII levels (Low = 30 mV, High = 100 mV) 010 Intermediate level (Low =38 mV, High = 120 mV) 011 Intermediate level (Low = 50 mV, High 150 mV) 100 Default PCI Express levels (Low = 65 mV, High = 175 mV) 101 Low = 75 mV, High = 200 mV 110 Intermediate level (Low = 88 mV, High = 225 mV) 111 Intermediate level (Low = 100 mV, High = 250 mV) *EICE9–8]:* 00 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs (application mode). 01 Exit from Idle ~88 UI and Unexpected Idle Detect ~10 µs 10 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs 11 Bypass
— 7–5	0	Reserved. Write to zero for future compatibility.	
EICF 4–0	0	**Lane F Receiver Idle Detection Control** The field is broken into two subfields. The highest order 3 bits set the general detection level. The lowest order 2 bits control exit from idle for the PCI Express interface. Recommended setting per protocol: • PCI Express: 10000. • SGMII: 00100. • Serial RapidIO: 00000.	*EICF[4–2]:* 000 Loss of signal detect function disabled. 001 Default SGMII levels (Low = 30 mV, High = 100 mV) 010 Intermediate level (Low =38 mV, High = 120 mV) 011 Intermediate level (Low = 50 mV, High 150 mV) 100 Default PCI Express levels (Low = 65 mV, High = 175 mV) 101 Low = 75 mV, High = 200 mV 110 Intermediate level (Low = 88 mV, High = 225 mV) 111 Intermediate level (Low = 100 mV, High = 250 mV) *EICF[1–0]:* 00 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs (application mode). 01 Exit from Idle ~88 UI and Unexpected Idle Detect ~10 µs 10 Exit from Idle ~88 UI and Unexpected Idle Detect ~1 µs 11 Bypass

15.8.32 SRDS Control Register 5 (SRDSnCR5)

SRDSnCR5 SRDS Control Register 5 Offset 0x14

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—						SDFMA		—						SDFMB	
Type	R/W															
Reset	0	0	0	0	0	0	x	x	0	0	0	0	0	0	x	x
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—					SDTXLA			—					SDTXLB		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR5 contains functional control bits for the SerDes interface logic.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-36 describes the SRDSnCR5 fields.

Note: Lane A corresponds to SerDes lane 0 and Lane B corresponds to SerDes lane 1.

Table 15-36. SRDSnCR5 Field Descriptions

Bits	Reset	Description		
— 31–26	0	Reserved. Write to zero for future compatibility.		
SDFMA 25–24	xx	**Lane A Digital Filter Bandwidth** Sets the bandwidth of the digital filter that optimizes for a given frequency offset. Recommended setting per protocol: • PCI Express: 01. • SGMII: 00. • Serial RapidIO: 00.	00 01 10 11	200 ppm (SGMII or Serial RapidIO protocol) 600 ppm (PCI Express) Reserved. Reserved.
— 23–18	0	Reserved. Write to zero for future compatibility.		
SDFMB 17–16	0000xx	**Lane B Digital Filter Bandwidth** Sets the bandwidth of the digital filter that optimizes for a given frequency offset. Recommended setting per protocol: • PCI Express: 01. • SGMII: 00. • Serial RapidIO: 00.	00 01 10 11	200 ppm (SGMII or Serial RapidIO protocol) 600 ppm (PCI Express) Reserved. Reserved.
— 15–11	0	Reserved. Write to zero for future compatibility.		

Table 15-36. SRDSnCR5 Field Descriptions (Continued)

Bits	Reset	Description		
SDTXLA 10–8	0	**Lane A Transmitter Amplitude Levels** Controls transmitter amplitude levels. Recommended setting per protocol: • PCI Express: 000. • SGMII: 000. • Serial RapidIO: 000.	000 001 010 011 100 101 110 111	No amplitude reduction. 0.916 full swing. 0.833 full swing. 0.750 full swing. 0.666 full swing. 0.583 full swing. 0.500 full swing. Reserved.
— 7–3	0	Reserved. Write to zero for future compatibility.		
SDTXLB 2–0	0	**Lane B Transmitter Amplitude Levels** Controls transmitter amplitude levels. Recommended setting per protocol: • PCI Express: 000. • SGMII: 000. • Serial RapidIO: 000.	000 001 010 011 100 101 110 111	No amplitude reduction. 0.916 full swing. 0.833 full swing. 0.750 full swing. 0.666 full swing. 0.583 full swing. 0.500 full swing. Reserved.

15.8.33 SRDS Control Register 6 (SRDSnCR6)

SRDSnCR6 SRDS Control Register 6 Offset 0x18

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—						SDFME		—						SDFMF	
Type	R/W															
Reset	0	0	0	0	0	0	x	x	0	0	0	0	0	0	x	x
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—						SDTXLE		—						SDTXLF	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: Reset values designated by x are determined by selected reset configuration.

Note: Always write reserved bits with the value they return when read. That is, to program the register, read the value, modify the appropriate fields, and then write back the resulting value.

SRDSnCR6 contains functional control bits for the SerDes interface logic.

Note: SerDes Port 1 uses a base address of 0xFFFAC000.
SerDes Port 2 uses a base address of 0xFFFAD000.

Table 15-37 describes the SRDSnCR6 fields.

Note: Lane E corresponds to SerDes lane 2 and Lane F corresponds to SerDes lane 3.

Table 15-37. SRDSnCR6 Field Descriptions

Bits	Reset	Description
— 31–26	0	Reserved. Write to zero for future compatibility.

Table 15-37. SRDSnCR6 Field Descriptions (Continued)

Bits	Reset	Description		
SDFME 25–24	xx	**Lane E Digital Filter Bandwidth** Sets the bandwidth of the digital filter that optimizes for a given frequency offset. Recommended setting per protocol: • PCI Express: 01. • SGMII: 00. • Serial RapidIO: 00.	00 01 10 11	200 ppm (SGMII or Serial RapidIO protocol) 600 ppm (PCI Express) Reserved. Reserved.
— 23–18	0	Reserved. Write to zero for future compatibility.		
SDFMF 17–16	0000xx	**Lane F Digital Filter Bandwidth** Sets the bandwidth of the digital filter that optimizes for a given frequency offset. Recommended setting per protocol: • PCI Express: 01. • SGMII: 00. • Serial RapidIO: 00.	00 01 10 11	200 ppm (SGMII or Serial RapidIO protocol) 600 ppm (PCI Express) Reserved. Reserved.
— 15–11	0	Reserved. Write to zero for future compatibility.		
SDTXLE 10–8	0	**Lane E Transmitter Amplitude Levels** Controls transmitter amplitude levels. Recommended setting per protocol: • PCI Express: 000. • SGMII: 000. • Serial RapidIO: 000.	000 001 010 011 100 101 110 111	No amplitude reduction. 0.916 full swing. 0.833 full swing. 0.750 full swing. 0.666 full swing. 0.583 full swing. 0.500 full swing. Reserved.
— 7–3	0	Reserved. Write to zero for future compatibility.		
SDTXLF 2–0	0	**Lane F Transmitter Amplitude Levels** Controls transmitter amplitude levels. Recommended setting per protocol: • PCI Express: 000. • SGMII: 000. • Serial RapidIO: 000.	000 001 010 011 100 101 110 111	No amplitude reduction. 0.916 full swing. 0.833 full swing. 0.750 full swing. 0.666 full swing. 0.583 full swing. 0.500 full swing. Reserved.

Serial RapidIO Controller 16

The RapidIO controller supports 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 RapidIO architecture provides a rich variety of features, including high data bandwidth, low-latency capability, and support for high-performance I/O devices. RapidIO technology provides message passing and software-managed programming models. The MSC8156E includes a serial RapidIO subsystem that supports two ports and a RapidIO message unit (RMU) that comply with the *RapidIO Interconnect Specification, Revision 1.2* (see **Figure 16-1**). The MSC8156E device can connect directly to a host, another MSC8156E device, or a serial RapidIO switch. Each port in the switch is point-to-point connected to the MSC8156E device through a serial RapidIO link that typically carries packets in both directions. Packets ready for processing are transported from the host to the MSC8156E, and the processed packets are transported from the MSC8156E device back to the host.

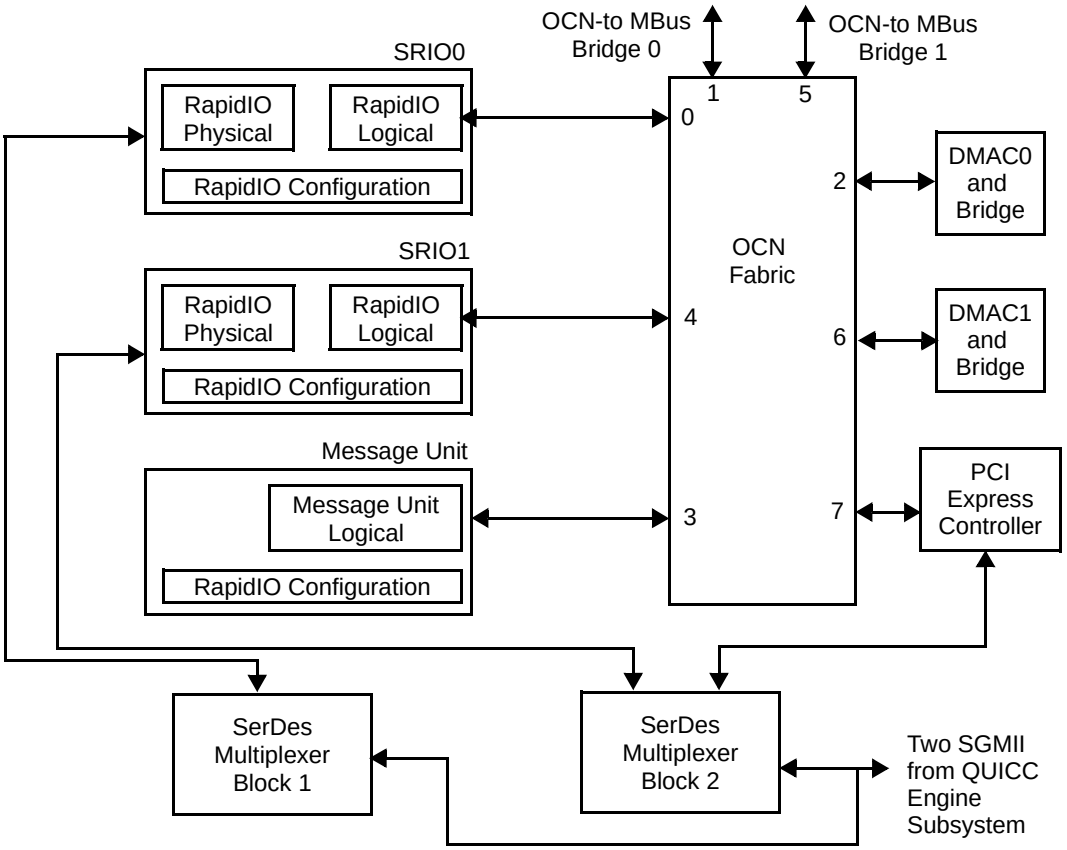
The Serial RapidIO controller directs the traffic flow between the MSC8156E and any other RapidIO device through the RMU for messages and doorbells and through the RapidIO DMA channels for NWRITEs, NWRITE_Rs, NREADs, and SWRITEs.

The two serial RapidIO ports connect through an OCN interconnect fabric in the HSSI (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*). This connectivity enables the two ports to perform pass-through operations between them. Pass-through mode allows RapidIO packets to be forwarded to the next device connected to the second RapidIO port. a packet is forwarded if the packet destination device ID does not match the current RapidIO port device ID (base and alternative, if enabled).

The host and the MSC8156E communicate as follows:

- The host sends messages to the destination MSC8156E device, which are sent back to the host after processing along with a short doorbell interrupt.
- Messages eliminate the latency of read accesses. The host writes to the MSC8156E and the MSC8156E writes to the host. In addition, messages can be used in applications where the host does not know the internal memory structure of the target DSP.
- The host can directly access the data in the MSC8156E memory for both reads and writes. It handshakes with the software running on a DSP core through buffer descriptors (BDs) that are messaged from the DSP core to the host.

- The host can put all the data buffers into its memory and have the MSC8156E access the data.
- Any initiator on the RapidIO system can access any internal memory space as well as the DDR SDRAM using NREAD, NWRITE, MESSAGE, and DOORBELLS. In addition, it can configure the RapidIO messaging and configuration unit using maintenance packets.
- The MSC8156E can perform NREAD, NWRITE, NWRITE_R, SWRITE, or MAINTENANCE accesses to any device directly connected to it, or to any other device that is part of the RapidIO interconnection through RapidIO switches. This capability is in addition to the MESSAGES and DOORBELL transactions already described.



Note: See **Chapter 15** *High Speed Serial Interface Unit* for details on the OCN fabric, bridges, and DMA operation.

Figure 16-1. RapidIO Controller and RMU

16.1 Introduction

This section summarizes the Serial RapidIO controller features, operating modes, and signal functionality.

16.1.1 Features

The RapidIO port incorporates the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- 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.
- Register and register bit extensions as described in the *RapidIO Interconnect Specification, Revision 1.2*, Part VIII: Error Management Extensions Specification.
- Packet types as defined in the *RapidIO Interconnect Specification, Revision 1.2*, Part II: Message Passing Logical Specification. Type 10 (send a short message) and Type 11 (send a message).
- NWRITE, SWRITE, and NWRITE_R write operations.
- Receiver-controlled flow control only.
- External message passing unit can handle inbound and outbound message passing, inbound and outbound doorbells, and inbound maintenance port-write operations.
- Inbound transactions to the configuration registers are limited to 32-bit accesses.
- Accepts and generates packet types as defined in the *RapidIO Interconnect Specification, Revision 1.2, Part II: Message Passing Logical Specification*. Specifically: Type 10 (send/receive a short message), and Type 11 (send/receive a message).
- Accepts and generates NREAD, NWRITE, SWRITE, and NWRITE_R, and MAINTENANCE read and write operations.
- Outbound maintenance transactions can be any valid size.

The RMU incorporates the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Two outbound message controllers.
- Two inbound message controllers.
- One outbound doorbell controller.
- One inbound doorbell controller.
- One inbound port-write controller.

The RMU incorporates the following general features of the RapidIO specification:

- Small or large size transport information field.
- All transaction flows and all priorities.
- Register and register bit extensions as described in Part VIII: Error Management Extensions Specification of the *RapidIO Interconnect Specification, Revision 1.2.*

The RMU supports the following user-defined features:

- Performance monitor interface.

The RapidIO endpoint incorporates the following user-defined features:

- Nine outbound ATMU windows.
- Five inbound ATMU windows.
- Logical outbound packet time-to-live counter to prevent local processor from hanging when the RapidIO interface fails.
- Accept-all mode of operation for failover support.
- RapidIO random bit error injection.
- Performance monitor interface for selected events.

RapidIO endpoint does not support or has limited support for the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- 50-bit and 66-bit addressing.
- Software assisted error recovery.
- ATOMIC transaction in either direction.
- Coherent (CC-NUMA) transactions.
- Transmitter-controlled flow control.
- No support for multicast event control symbols.

The RapidIO endpoint incorporates the following features of RapidIO 1x/4x LP-serial interfaces:

- Both 1x and 4x LP-serial link interfaces.
- Transmission rates of 1.25, 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps, respectively).
- Auto detection of 1x and 4x mode operation during port initialization.
- Error detection for packets and control symbols.
- Link initialization, synchronization, error recovery, and time-out.

RapidIO endpoint does not support the following features of RapidIO 1x/4x operation:

- RapidIO endpoint cannot be configured as four 1x ports.

16.1.2 Operating Modes

The main operating modes of the RapidIO ports are as follows:

- 1x or 4x LP-Serial link interfaces.
- Transmission rates of 1.25. 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps, respectively).
- Small or large size transport information field.
- Accept-all mode of operation; all packets are accepted regardless of the target ID.

The main operating modes of the message unit are as follows:

- Outbound message controllers:
  - — *Direct mode*. There are no descriptors, and software must initialize the required fields before starting a transfer.
  - — *Chaining mode*. Software must initialize descriptors in memory and the required fields before starting a transfer.
  - — *Multicast mode*. Single-segment messages can be transferred up to 32 destinations.
- Inbound message controllers:
  - — *Direct mode*. There are no descriptors, and software must initialize the required fields before starting a transfer.

16.1.3 1x/4x LP-Serial Signals

Table 16-1 describes the serial RapidIO signal functionality. For details on electrical characteristics, refer to the *RapidIO Interconnect Specification, Revision 1.2*.

Table 16-1. Serial Rapid I/O Interface Signals

Port Name	Description	System Connection	I/O	Active State	Reset
SD_TX[0–3]/ SD_TX[0–3]	**Transmit Data** Serial data output for the 1x or 4x link. One differential pair output per link. This implementation supports data rates of 1.25, 2.5, or 3.125 Gbaud.	Primary output pads. Asynchronous outputs.	O	—	x
SD_RX[0–3]/ SD_RX[0–3]	**Receive Data** Serial data input for 1x or 4x link. One differential pair input per link. This implementation supports data rates of 1.25, 2.5, or 3.125 Gbaud.	Primary input pads. Asynchronous inputs.	I	—	x

16.1.4 RapidIO Interface Activation

There are two basic activation scenarios:

- Boot over the Serial RapidIO interface
- Non-boot operation

16.1.4.1 Initialization for Booting the MSC8156E DSP

When the RapidIO interface is use to boot the MSC8156E, the serial RapidIO master device waits for the boot program to complete the default initialization and then initializes the interface by loading code and data into the device memory. For details, see **Section 6.2.4**, *Serial RapidIO Interconnect*, on page 6-21. In this scenario, the boot operation opens the lanes before the 4x sync sequence ends and the master device can begin accessing the RapidIO interface.

16.1.4.2 Initialization for Non-Boot Operation

When used for non-boot operations, the interface is connected to a serial RapidIO transmitting device, such as an MSBA8100, MSC8144, or another MSC8156E, for example. Sync signals are received over the Rx lines, although the Tx lines remain tri-stated. The device is unable to sync in 4x mode and times out because the Tx lines are tri-stated, and the interface syncs on 1x mode. To sync on 4x mode, the device must be disabled, by writing 0x50E00001 to it, which activates the lines, removing them from tri-state, and reactivates the device in 4x mode.

16.2 RapidIO Interface Basics

This section summarizes the RapidIO transactions, packet format, and control symbols. It also discusses how the configuration registers are accessed via the RapidIO packets and the operation of the ATMU translation windows for translating RapidIO addresses to local physical addresses and vice versa.

16.2.1 RapidIO Transactions

The RapidIO endpoint supports limited inbound and outbound RapidIO transactions and all RapidIO message passing transactions, as listed in **Table 16-2**.

Table 16-2. RapidIO I/O Inbound Transactions

RapidIO Transaction	ftype	ttype	Status	Description
NREAD	0010	0100	NA	Read
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-write[1]
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

Notes: **1.** Limited to inbound RapidIO packets only.

Table 16-3 summarizes the RapidIO message passing transactions of the RapidIO endpoint

Table 16-3. RapidIO Message Passing Transactions

Message 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

16.2.2 RapidIO Packet Format

Table 16-4 summarizes the small transport field packet formats of RapidIO transaction types for LP-Serial operation.

Note: RapidIO endpoint limits configuration read and write requests to 32-bit data accesses. The large transport field packet format extends the destination and source IDs to 16-bits each. The MSC8156E supports small and large transport fields (large at default), so, for large transport, the destination and source IDs are 16-bits wide according to the direction of the transaction.

Table 16-4. RapidIO Small Transport Field Packet Format

Transaction	Bits															
	32							32					16			64
	5	3	2	2	4	8	8	4	4	8	8	8	13	1	2	
NREAD	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	addr			wd ptr	xambs	NA
NWRITE_R,	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	addr			wd ptr	xambs	dword 0 → dword n
NWRITE	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	wrsize	don't care	addr			wd ptr	xambs	dword 0 → dword n
SWRITE	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	addr(29), rsv(1)=0, xambs(2)					dword 0 -> dword n			
MAINT read	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	rd/wr size	src TID	hop cnt	cfg offset		wd ptr	rsv = 0	NA
MAINT write	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	ttype	rd/wr size	src TID	hop cnt	cfg offset		wd ptr	rsv = 0	dword 0 (32-bit)
MAINT port-write	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	rd/wr size	rsv=0	hop cnt	rsv=0		wd ptr	rsv = 0	dword 0 → dword n
MAINT response without data	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				NA

Table 16-4. RapidIO Small Transport Field Packet Format (Continued)

Transaction	Bits															
	32							32					16			64
	5	3	2	2	4	8	8	4	4	8	8	8	13	1	2	
MAINT response with data	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	type	status	tar TID	hop cnt	rsv=0				dword 0 (32-bit)
RESPONSE without data	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	type	status	tar TID	NA					
RESPONSE without data for message	ackID	rsv = 0	prio	tt	ftype	dest ID	src ID	type	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	type	status	tar TID	dword 0 -> dword n					
DOORBELL	ackID	rsv =0	prio	tt	ftype	destID	src ID	rsv=0		src TID	Info-msb	Info - lsb	NA			
MESSAGE	ackID	rsv =0	prio	tt	ftype	dest ID	src ID	msglen(4), ssize(4), letter(2), mbox(2), msgseg(4)			dword 0 -> dword n					

16.2.3 RapidIO Control Symbol Summary

Table 16-5 summarizes the 1x/4x LP-Serial control symbols and their format. Refer to the *RapidIO Interconnect Specification, Revision 1.2* Part VI: Physical Layer 1x/4x LP-Serial Specifications, Chapter 4 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 composed of the 8-bit special character and the 24-bit control symbol format.

Table 16-5. 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

Table 16-5. 1x/4x LP-Serial Control Symbol Format (Continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
010	pkt_ackID	cause	—		crc	Packet not accepted Cause: 00001 Received unexpected ackID on packet. 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 00010 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 16-5. 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
—			111	000	crc	NOP (ignore)

16.2.4 Accessing Configuration Registers via RapidIO Packets

The RapidIO endpoint limits requests to configuration register space to 32-bit data accesses. If the order of completion is important, inbound configuration accesses should be assumed incomplete until an appropriate response is received. There should be only one outstanding configuration request at a time to ensure that requests complete in the intended order.

16.2.4.1 Inbound Maintenance Accesses

There are two recommended methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

One 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 (LCSxBA1CSR), which is described on **page 16-143**. If external configuration accesses are disabled (LLCR[ECRAB] = 1; see **page 16-174**), any configuration access through the LCSxBA1CSR window is denied. A 32-bit data payload of all zeros is returned for a non-maintenance configuration read.

Note: Only NWRITE_R requests can access the entire internal CCSR address space. Any other write transaction is denied and does not alter the registers.

The second method is based on RapidIO MAINT requests. This method allows an external device limited access to local RapidIO configuration register space. Any maintenance access beyond the first 64 KB of RapidIO configuration register space is denied (lower 64 KB contains RapidIO architecture registers; upper 64 KB contains RapidIO implementation registers). A 32-bit data payload of all zeros is returned for a read response.

A third method that uses an inbound ATMU window to translate RapidIO NREAD and NWRITE_R requests to configuration accesses is not recommended because it does not support

the configuration access protection features offered by the LCSBA1CSR window and RapidIO MAINT requests.

16.2.4.2 RapidIO Non-Maintenance Accesses Using LCSBA1CSR

NREAD and NWRITE_R requests can be used to access RapidIO configuration register space by matching RapidIO address[0–13] with the 14-bit value of the LCSBA1CSR[LCSBA], which is described on **page 16-143**. Inbound requests with RapidIO addresses that fall within the window defined by LCSBA1CSR are translated to the local address range indicated by the CCSR base address. The LCSBA1CSR hit definition and RapidIO address translation depend on the size of the configuration register space, which is fixed at 1 MB.

The LCSBA1CSR hit definition is:

- A window hit is defined as LCSBA1CSR[30–17] matching RapidIO address [0–13].
- The accessed RapidIO register offset is derived the RapidIO address[14–30].

If the NREAD/NWRITE_R access hits an inbound ATMU window as well, the LCSBA1CSR window has priority in determining the RapidIO address translation. If an NWRITE/SWRITE request hits the LCSBA1CSR window, an illegal transaction decode error is generated and logged.

16.2.4.3 RapidIO Maintenance Accesses

MAINT requests can be used to access RapidIO configuration register space via the 21-bit configuration offset field (config_offset) from the RapidIO maintenance packet. The index into RapidIO configuration register space is represented by config_offset[8–20] only; bits for config_offset[0–7] are ignored.

16.2.4.3.1 Guidelines

The RapidIO endpoint limits configuration register space requests to 32-bit data accesses. If the order of completion is important, assume that inbound configuration accesses are incomplete until an appropriate response is received. It is suggested that only one outstanding configuration request be active at a time to ensure that requests are completed in the intended order. For inbound configuration write results that are immediately used by another transaction, perform an inbound configuration read immediately after the configuration write to ensure that the transaction uses the updated value of the accessed configuration register.

16.2.4.3.2 Outbound Maintenance Accesses

Outbound NREAD_R or NWRITE requests can be translated to a RapidIO maintenance request if the internal generated 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, and priority for the outbound RapidIO packet.

16.2.5 RapidIO ATMU Implementation

The ATMU uses a set of registers to translate RapidIO packets to internal packets on inbound and to translate internal packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits result in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4 K and the largest window size is 16 G for inbound translation and 64 G for outbound translation.

The default window register set causes no translation of the transaction address for inbound transactions because the RapidIO address space has 34 bits and the internal interconnect address space has 36 bits. For outbound transactions, the default window maps each of the four 16 G windows to the RapidIO 16 G 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 36-bit (0–35) internal interconnection address. The MSC8156E is confined to 32-bit internal addresses, therefore the top 4 bits (0–3) of the Inbound translation address and the outbound base address should be set to all 0; setting any of these bits results in undefined behavior.

As with all registers, an external processor writing the ATMU registers should never assume that the write is completed until a response is received.

Note: Booting from a serial RapidIO must always use outbound ATMU window 0.

16.2.5.1 RapidIO Outbound ATMU

All outbound windows have the capability to have 2 or 4 segments, all of which are equal in size, numbered 0–1, or 0–3, 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: 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 16-2** shows an example of this capability. A window is defined to be 4 Kbyte 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.

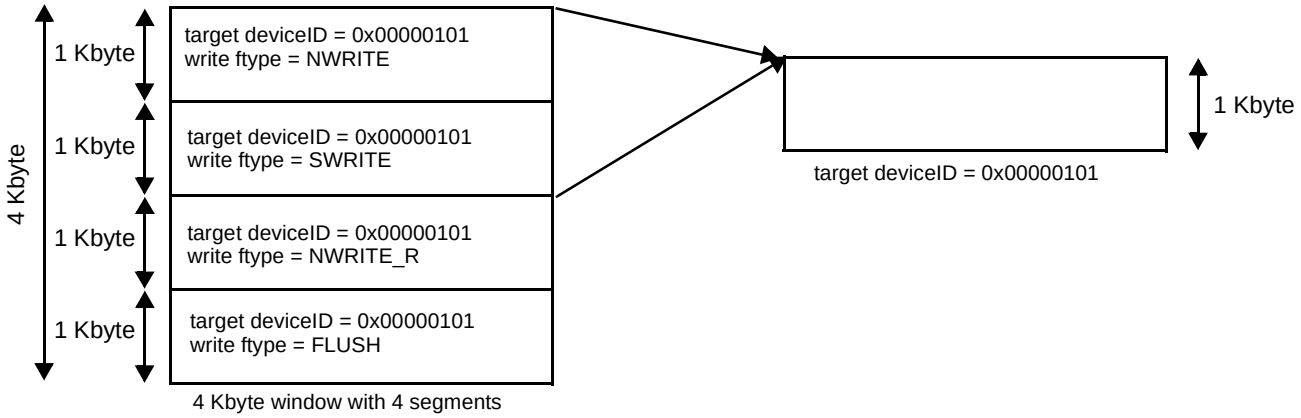


Figure 16-2. Single Target Example

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 16-3** 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 type 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.

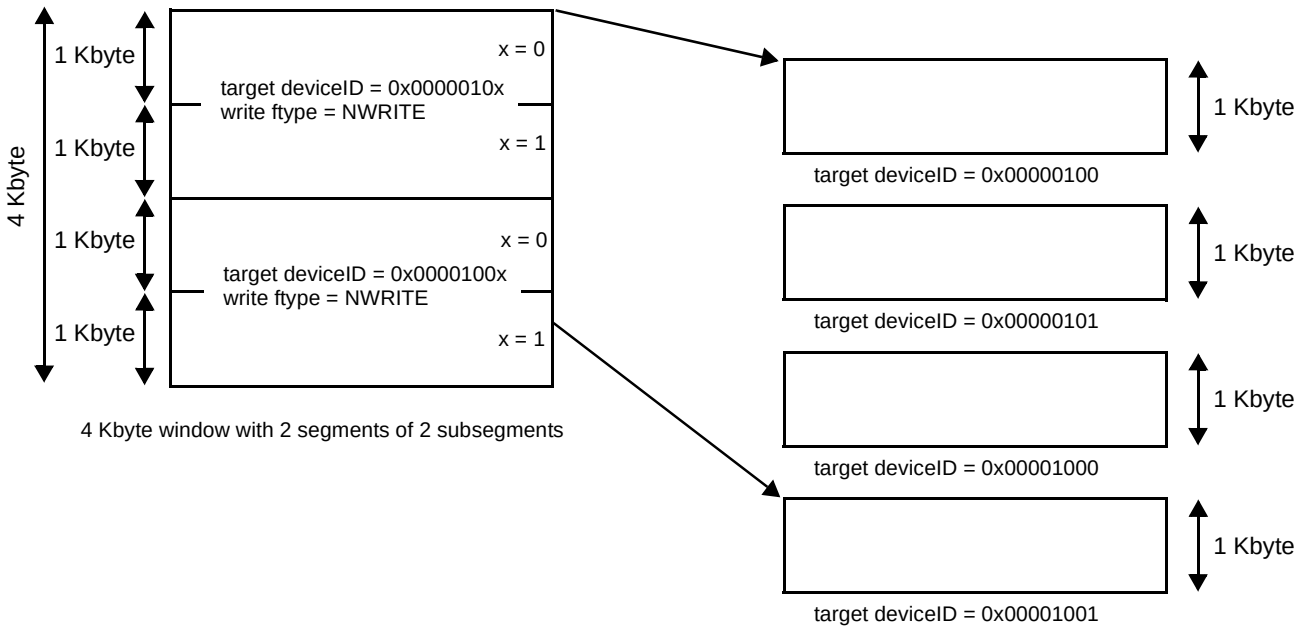


Figure 16-3. Multi-Targeting Example

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.

16.2.5.2 Outbound Windows

RapidIO Endpoint implements nine outbound ATMU translation windows for translating local physical address to RapidIO address.

- Port n RapidIO Outbound Window Translation Address Registers 0–8 define the starting point for the RapidIO address translation and specify the RapidIO destination ID for the transaction.
- Port n RapidIO Outbound Window Attributes Registers 0–8 define the translation window size and specify the RapidIO transaction type and priority for the transaction.
- Port n RapidIO Outbound Window Segment 1–3 Registers 1–8 are used if the ATMU is segmented.

There are eight comparison windows.

- Port n RapidIO Outbound Window Base Address Register 1–8 represents the base address for each of the eight ATMU windows. The base address for each window must be aligned based on the translation window size specified in the Port n RapidIO Outbound Window Attributes Register 1–8.

- Port n RapidIO Outbound Window Translation Address Register 0 and Port n RapidIO Outbound Window Attributes Register 0 are the translation registers for the default ATMU window. It is used only if an NREAD, NWRITE, or NWRITE_R request misses all eight ATMU comparison windows.

16.2.5.3 Window Size and Segmented Windows

For the following discussion, RapidIO address[0–30] consists of xambs[0–1] and address[0–28] fields as defined in the RapidIO request packet format. RapidIO address is a 31-bit double-word physical address (or 34-bit byte address). Internal address[0–32] is a 33-bit double-word physical address (or 36-bit byte address).

The ATMU window hit definition and RapidIO address translation are as follows:

1. 4K window size (smallest window size)
   - A window hit is defined as {BEXADD[0–3], BADD[0–19]} matching internal address [0–23]
   - RapidIO addr[0–30] = {TREXAD[8–9], TRAD[0–19], internal address[24–32]}
2. 8K window size
   - A window hit is defined as {BEXADD[0–3], BADD[0–18]} matching internal address [0–22]
   - RapidIO addr[0–30] = {TREXAD[8–9], TRAD[0–18], internal address[23–32]}
3. 16K window size
   - A window hit is defined as {BEXADD[0–3], BADD[0–17]} matching internal address [0–21]
   - RapidIO addr[0–30] = {TREXAD[8–9], TRAD[0–17], internal address[22–32]}
4. Window sizes 32 K, 64 K, 128 K, 256 K, 512 K, 1 M, 2 M, 4 M, 8 M, 16 M, 32 M, 64 M, 128 M, 256 M, 512 M, 1 G, and 2 G are not shown
5. 4G window size
   - A window hit is defined as {BEXADD[0–3]} matching internal address [0–3]
   - RapidIO addr[0–30] = {TREXAD[8–9], internal address[4–32]}
6. 8G window size
   - A window hit is defined as {BEXADD[0–2]} matching internal address [0–2]
   - RapidIO addr[0–30] = {TREXAD[8], internal address[3–32]}
7. 16 G window size
   - A window hit is defined as {BEXADD[0–1]} matching internal address [0–1]
   - RapidIO addr[0–30] = {internal address[2–32]}

A window can be defined to be non-segmented or segmented depending on the NSEG field definition in the Port n RapidIO Outbound Window Attributes Register.

- A non-segmented window uses the specified RapidIO transaction type (RDTYP, WRTYP) and designated target ID (TGTID) without additional internal address comparison.
- A segmented window divides the specified window size into smaller sub-windows. A segmented window can be further divided into sub-segments as defined by the NSSEG field definition. The use of segmented and sub-segmented windows requires additional internal address comparison. There are two reasons for using a segmented ATMU window: allow a single ATMU window to generate different transactions types; allow a single ATMU window to generate multiple RapidIO target IDs.

Table 16-6 lists the various combination options.

Table 16-6. Outbound ATMU Window Segments

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
1	0	1	NA	PnROWTAR0 [TREXAD]	PnROWTAR0 [TREXAD]	PnROWAR0
2	0	1	NA	[7–0] = PnROWTAR0 {TREXAD]	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–0] = PnROWTAR0 [TREXAD]	PnROWAR0
		2	NA	[7–0] = PnROWS1R1 [SGTGTDID]	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–0] = PnROWS1R1 [SGTGTDID]	PnROWS1R1

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	0	1	NA	[7–0] = PnROWTAR0 [TREXAD]	[15–10] = PnROWTEAR0 [LTGTID], [9:8] = PnROWTAR0 [LTGTID], [7–0] = PnROWTAR0 [TREXAD]	PnROWAR0
		2	NA	[7–0] = PnROWS1R1 [SGTGTDID]	[15–10] = PnROWTEAR0 [LTGTID], [9:8] = PnROWTAR0 [LTGTID], [7–0] = PnROWS1R1 [SGTGTDID]	PnROWS1R1
		3	NA	[7–0] = PnROWS2R1 [SGTGTDID]	[15–10] = PnROWTEAR0 [LTGTID], [9:8] = PnROWTAR0 [LTGTID], [7–0] = PnROWS2R1 [SGTGTDID]	PnROWS2R1
		4	NA	[7–0] = PnROWS3R1 [SGTGTDID]	[15–10] = PnROWTEAR0 [LTGTID], [9:8] = PnROWTAR0 [LTGTID], [7–0] = PnROWS3R1 [SGTGTDID]	PnROWS3R1
1	2	Not supported	Not supported	Not supported	Not supported	Not supported

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	2	1	1	[7–1] = PnROWTAR0 [TREXAD], 0 = 0b0	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWTAR0 [TREXAD], 0 = 0b0	PnROWAR0
			2	[7–1] = PnROWTAR0 [TREXAD], 0 = 0b1	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWTAR0 [TREXAD], 0 = 0b1	
		2	1	[7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b0	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b0	PnROWS1R1
			2	[7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b1	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b1	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	4	1	1	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b00	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b00	PnROWAR0
			2	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b01	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b01	
			3	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b10	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b10	
			4	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b11	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b11	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	4	2	1	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b00	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b00	PnROWS1R1
			2	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b01	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b01	
			3	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b10	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b10	
			4	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b11	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b11	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	8	1	1	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b000	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b000	PnROWAR0
			2	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b001	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b001	
			3	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b010	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b010	
			4	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b011	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b011	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	8	1	5	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b100	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b100	PnROWAR0
			6	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b101	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b101	
			7	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b110	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b110	
			8	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b111	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b111	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	8	2	1	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b000	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b000	PnROWS1R1
			2	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b001	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b001	
			3	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b010	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b010	
			4	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b011	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b011	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
2	8	2	5	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b100	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b100	PnROWS1R1
			6	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b101	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b101	
			7	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b110	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b110	
			8	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b111	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b111	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	2	1	1	[7–1] = PnROWTAR0 [TREXAD], 0 = 0b0	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWTAR0 [TREXAD], 0 = 0b0	PnROWAR0
			2	[7–1] = PnROWTAR0 [TREXAD], 0 = 0b1	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWTAR0 [TREXAD], 0 = 0b1	
		2	1	[7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b0	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b0	PnROWS1R1
			2	[7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b1	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS1R1 [SGTGTDID], 0 = 0b1	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	2	3	1	[7–1] = PnROWS2R1 [SGTGTDID], 0 = 0b0	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS2R1 [SGTGTDID], 0 = 0b0	PnROWS2R1
			2	[7–1] = PnROWS2R1 [SGTGTDID], 0 = 0b1	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS2R1 [SGTGTDID], 0 = 0b1	
		4	1	[7–1] = PnROWS3R1 [SGTGTDID], 0 = 0b0	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS3R1 [SGTGTDID], 0 = 0b0	PnROWS3R1
			2	[7–1] = PnROWS3R1 [SGTGTDID], 0 = 0b1	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–1] = PnROWS3R1 [SGTGTDID], 0 = 0b1	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	4	1	1	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b00	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b00	PnROWAR0
			2	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b01	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b01	
			3	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b10	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b10	
			4	[7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b11	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWTAR0 [TREXAD], [1–0] = 0b11	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	4	2	1	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b00	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b00	PnROWS1R1
			2	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b01	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b01	
			3	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b10	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b10	
			4	[7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b11	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS1R1 [SGTGTDID], 0 = 0b11	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	4	3	1	[7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b00	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b00	PnROWS2R1
			2	[7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b01	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b01	
			3	[7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b10	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b10	
			4	[7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b11	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS2R1 [SGTGTDID], 0 = 0b11	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	4	4	1	[7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b00	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b00	PnROWS3R1
			2	[7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b01	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b01	
			3	[7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b10	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b10	
			4	[7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b11	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–2] = PnROWS3R1 [SGTGTDID], 0 = 0b11	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	1	1	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b000	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b000	PnROWAR0
			2	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b001	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b001	
			3	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b010	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b010	
			4	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b011	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b011	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	1	5	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b100	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b100	PnROWAR0
			6	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b101	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b101	
			7	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b110	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b110	
			8	[7–3] = PnROWTAR0 [TREXAD], 0 = 0b111	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWTAR0 [TREXAD], 0 = 0b111	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	2	1	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b000	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b000	PnROWS1R1
			2	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b001	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b001	
			3	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b010	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b010	
			4	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b011	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b011	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	2	5	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b100	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b100	PnROWS1R1
			6	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b101	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b101	
			7	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b110	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b110	
			8	[7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b111	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS1R1 [SGTGTDID], 0 = 0b111	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	3	1	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b000	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b000	PnROWS2R1
			2	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b001	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b001	
			3	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b010	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b010	
			4	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b011	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b011	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	3	5	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b100	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b100	PnROWS2R1
			6	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b101	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b101	
			7	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b110	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b110	
			8	[7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b111	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS2R1 [SGTGTDID], 0 = 0b111	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	4	1	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b000	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b000	PnROWS3R1
			2	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b001	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b001	
			3	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b010	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b010	
			4	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b011	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b011	

Table 16-6. Outbound ATMU Window Segments (Continued)

Desired Configuration		Registers That Need Configuration		Target ID		Transaction Type
# of Segments (NSEG)	# of Sub-Segments (NSSEG)	Segment Index (1–8	Subsegment Index (1–8)	Small Transport (7–0)	Large Transport (15–0)	
4	8	4	5	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b100	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b100	PnROWS3R1
			6	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b101	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b101	
			7	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b110	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b110	
			8	[7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b111	[15–10] = PnROWTEAR0 [LTGTID], [9–8] = PnROWTAR0 [LTGTID], [7–3] = PnROWS3R1 [SGTGTDID], 0 = 0b111	

The use of segmented windows impacts only the RapidIO transaction type or the destination ID. The internal address translation is a function of the Port n Outbound Window Translation Address Register and the translation window size.

16.2.5.3.1 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 highest 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 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 (1–8) and the transaction 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.

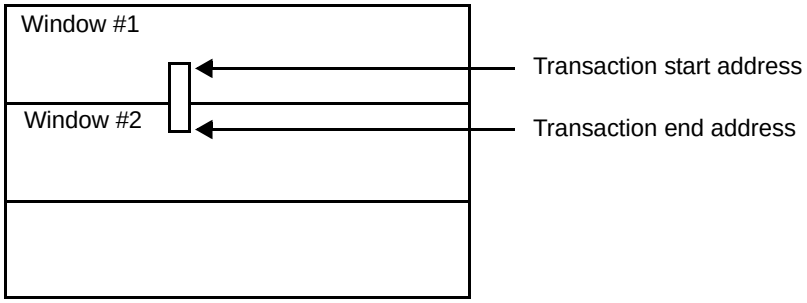


Figure 16-4. Valid Hit that Extends Into a Lower Priority Window

2. If a request hits (base address match) multiple ATMU windows (1–8) and the transaction 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.

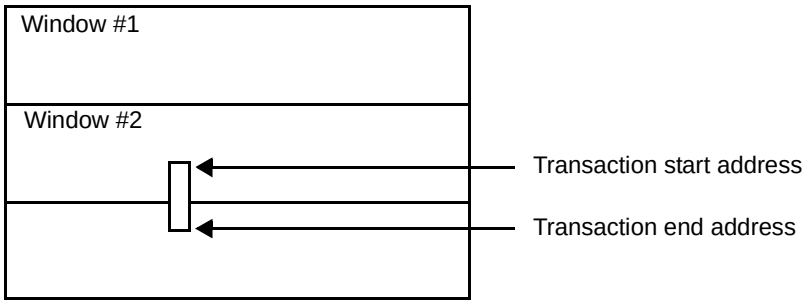


Figure 16-5. Valid Hit that Extends Beyond the Window Boundary

16.2.5.3.2 Window Boundary Crossing Errors

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 (1–8, default) and the transaction end address extends into another ATMU window with higher priority, an ATMU crossed boundary error is generated and logged. The outbound request is discarded.

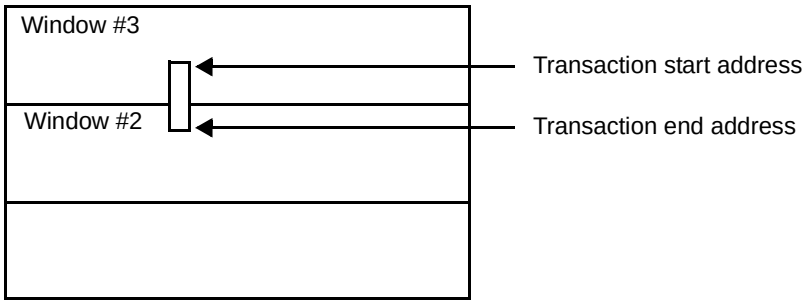


Figure 16-6. Boundary Crossing Error Due to Extension Into a Higher Priority Window

2. If a request hits multiple ATMU windows (1–8, default) and transaction 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.

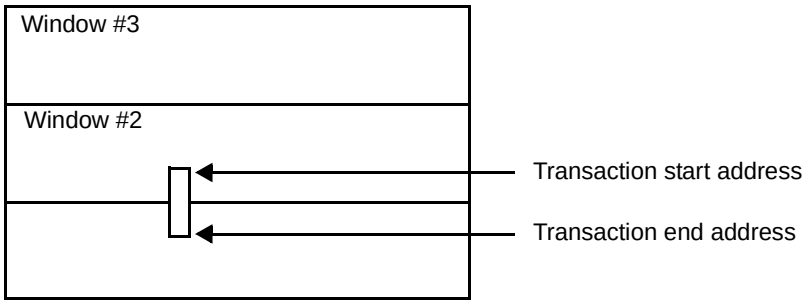


Figure 16-7. Boundary Crossing Error Due to Extension Beyond the Higher Priority Window Boundary

3. If a request hits (base address match) an ATMU window (1–8) and the transaction end address exceeds the size of the window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.

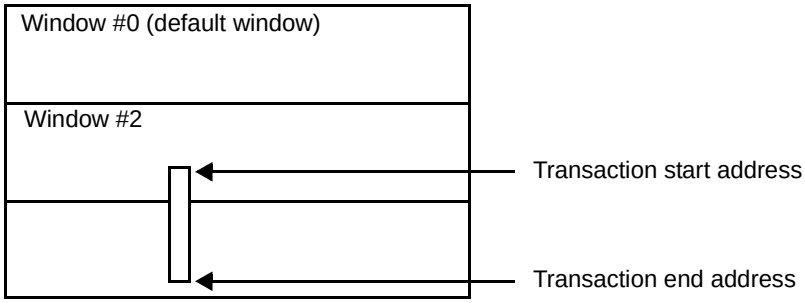


Figure 16-8. Boundary Crossing Error Due to Transaction Size Exceeding the Window Size

An internal error response is generated for internal requests that require a response. Boundary crossing errors (outbound ATMU boundary crossing, segment boundary crossing, and sub-segment boundary crossing) are logged in the LTLEDCSR[OACB] configuration register field. If a request misses all ATMU windows (1–8) and the transaction's end address exceeds the maximum size of the default window, an outbound ATMU crossed boundary error is not generated. The outbound request is forwarded to the RapidIO target device.

16.2.5.4 RapidIO Inbound ATMU

The RapidIO endpoint has five inbound ATMU translation windows for translating RapidIO addresses to local physical addresses. ATMU registers are used for inbound transactions. Their purpose is to translate RapidIO packets to on-device interconnect packets. ATMU window misses use the window 0 register set by default, and overlapping window matches result in the use of the lowest-number window register set in the match. For inbound translation, the smallest window size is 4 KB and the largest window size is 16G. The default window register set causes no translation of the transaction address for inbound transactions. The inbound translation windows must be aligned on the basis of 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 36-bit (0–35) internal addressing. The MSC8156E device is confined to 32-bit addresses, so the top 4 bits (0–3) of the inbound translation address should be set to all 0s. Other settings result in undefined behavior. An external processor should not assume that a write to any ATMU register is complete until a response is received. **Table 16-7** describes the registers for configuring the window parameters, along with the number of the page where each register is described in detail.

Table 16-7. ATMU Registers for Configuring Window Parameters

ATMU Registers	Acronym	Description	Page
Port 0–1 RapidIO Inbound Window Translation Address Registers 1–4	PnRIWTAR[1–4]	Define the starting-point for the RapidIO address translation.	**page 16-191**
Port 0–1 RapidIO Inbound Window Attributes Registers 1–4	PnRIWAR[1–4]	Define the translation window size and specify the internal priority, attributes, and the internal target port for the transaction.	**page 16-193**
Configuring the Four Comparison Windows			
The Port 0–1 RapidIO Inbound Window Base Address Registers 1–4	PnRIWBAR [1–4]	Represent the base address for each ATMU window. The base address must be aligned based on the translation window size specified in PnRIWAR 1–4	**page 16-193**
Port 0–1 RapidIO Inbound Window Translation Address Registers 0	PnRIWTAR0	Translation registers for ATMU window 0 (default window). Used for the following conditions: • NREAD, NWRITE_R request misses all four ATMU comparison windows and the LCSBA1CSR window. • NWRITE, SWRITE request misses all four comparison windows.	**page 16-191**
Port 0–1 RapidIO Inbound Window Attributes Register 0	PnRIWAR0		**page 16-193**

For the following discussion, RapidIO address[0–30] consists of xambs[0–1] and the address[0–28] fields as defined in the RapidIO request packet format. The RapidIO address is a 31-bit double-word physical address (or a 34-bit byte address). The ATMU translated address[0–32] is a 33-bit double-word physical address (or 36-bit byte address).

The ATMU window hit definition and RapidIO address translation are as follows:

- 4 KB window size (smallest window size):
  - A window hit is defined as {BEXADD[0–1], BADD[0–19]} matching RapidIO address [0–21].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], TRAD[0–19], RapidIO address[22–30]}.
- 8 KB window size:
  - A window hit is defined as {BEXADD[0–1], BADD[0–18]} matching RapidIO address [0–20].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], TRAD[0–18], RapidIO address[21–30]}.
- 16 KB window size:
  - A window hit is defined as {BEXADD[0–1], BADD[0–17]} matching RapidIO address [0–19].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], TRAD[0–17], RapidIO address[20–30]}.

- Window sizes 32 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB, 1GB, 2 GB are not shown.
- 4 GB window size:
  - A window hit is defined as {BEXADD[0–1]} matching RapidIO address [0–1].
  - Internal interconnection addr[0–32] = {TREXAD[0–3], RapidIO address[2–30]}.
- 8 GB window size:
  - A window hit is defined as {BEXADD[0]} matching RapidIO address0.
  - Internal interconnection addr[0–32] = {TREXAD[0–2], RapidIO address[1–30]}.
- 16 GB window size (largest size):
  - A window hit is defined as any RapidIO address.
  - Internal interconnection addr[0–32] = {TREXAD[0–1], RapidIO address[0–30]}.

16.2.5.4.1 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 has the next highest priority, 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 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.

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 this is not a practical programming application, RapidIO Endpoint handles it as follows:

- If a request hits (base address match) an ATMU window (1–4) and the transaction end address extends into another ATMU window with a lower priority but 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 (1–4) and the transaction 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.

16.2.5.4.2 Window Boundary Crossing Errors

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 situation as follows:

- If a request hits (base address match) an ATMU window (1–4, default) and the transaction end address extends into another ATMU window with a higher priority, an ATMU crossed boundary error is generated and logged.

- If a request hits multiple ATMU windows (1–4, default) and the transaction 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 as follows:

- If a request hits (base address match) an ATMU window (1–4) and the transaction end address exceeds the size of the window, an inbound ATMU crossed boundary error is generated and logged.
- If a NREAD/NWRITE_R/NWRITE/SWRITE request hits (base address match) an ATMU window (1–4, default) and the transaction 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. If a request misses all ATMU windows (1–4) and the transaction end address exceeds the maximum size of the default window, an inbound ATMU crossed boundary error is not generated.

16.2.6 Generating Link-Request/Reset-Device

In LP-Serial mode, the link partner cannot be reliably reset using the link-request/reset-device control symbols because the input port receiver cannot be disabled independently of the output port driver. The input port driver must 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 is received on the inbound side after one of the four link-request/reset-device control symbols is sent outbound, the required sequence of four link-request/reset-device symbols is interrupted with the packet acknowledgement (packet accept, packet retry, or packet not accept).

One method to reset the link partner is as follows.

1. Disable packet reception and transmission by setting the Port Lockout bit (PL bit = 1 in the Port n Control Command and Status Register).
2. Wait the appropriate amount of time for all outstanding packets transmitted on the link to be either retried, accepted or timed-out
3. Discard all outbound packets (OBDEN = 1 in Port n Physical Configuration Register) and clear all errors
4. Disable the input port receiver (IPD bit = 1 in the Port n Control Command and Status Register). This will allow the four consecutive link-request/reset-device control symbols to be generated with only idle control symbols between the link-request/reset-device control symbols.

5. Generate four link-request/reset-device control symbol using the Port n Link Maintenance Request register. Note that the link partner does not generate a link-response control symbol for a link-request/reset-device control symbol.
6. The link partner will expects the inbound and outbound Ack IDs to be 0 after being reset. Set the inbound and outbound Ack IDs to 0 (IA and OBA) by writing 0s to these fields in the Port n Local AckID Status Command and Status Register (PnLASCSR).
7. After the link partner has completed initialization indicated by the PO bit of Port n Error and Status Command and Status Register (PnESCSR), enable packets to be received and transmitted by clearing the Port Lockout bit (PL) in the Port n Control Command and Status Register (PnCCSR).

16.2.7 Outbound Drain Mode

The RapidIO port is placed into Drain mode when one of the following occurs:

- PnPCR[OBDEN] is set.
- he Failed Threshold has been encountered and the PnCCSR[SPF] and PnCCSR[DPE] are both set
- The packet time-to-live counter expires causing PnPCR[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 (because 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 were accepted or not. If the outbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of Drain mode. Also, if the link-partner needs to be returned to the inbound OK state, software should send a link-request/input-status. The recommended sequence for recovering from Drain mode is given in **Section 16.2.9**, *Software Assisted Error Recovery Register Support*, on page 16-47.

Setting PnPCR[OBDEN] also causes any queued packet acknowledgements to be discarded if the port is uninitialized; the RapidIO port allows 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 will occur if the packet response timer is enabled (PRTOCCSR is non 0).

16.2.8 Input Port Disable Mode

When PnCCSR[PD] is set, the RapidIO port is placed into Input Port Disable mode. This mode causes the following to occur:

- The RapidIO port discards all incoming data streams.
- Because the incoming stream is invalid, the RapidIO port returns it inbound port to the normal state. If an incoming packet is in progress when the RapidIO port is placed into Input Port Disable mode, the RapidIO port physical layer aborts the packet to the logical layer.
- The RapidIO port ends any packet capture that is in progress when the mode is invoked.
- The RapidIO port clears the link-request/reset-device count.
- The RapidIO port inbound ackID shows that all packets successfully received by the port before it entered Input Port Disable mode were accepted. If the outbound port entered Output Port Disable mode at the same time, however, some packet acknowledgements may be discarded by the RapidIO port. If the inbound ackID is not accepted, software should change it before taking the RapidIO port out of Input Port Disable mode.

16.2.9 Software Assisted Error Recovery Register Support

PnLMREQCSR is only supported for recovery from Drain mode. Therefore, software should only write to 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:
   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: If software can guarantee that the link-partner will not attempt to forward any packets to this RapidIO port, then software can 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 is already lined up, then the write actually causes the ackIDs not to match, and the link is not recovered.

5. Software should cause the link-partner to send a link-request/input-status to ensure that the RapidIO inbound port is operating normally.
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 PnLMREQCSR, software should make sure that PnLMRESPCSR successfully reads as set; otherwise, software may read a stale ackID status/link status later.

Note: When the RapidIO port's outbound ackID is written by software using PnLASCSR, 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, PnCCSR[PL] could be set to prevent the inbound ackID from changing before PnLASCSR is written.

16.2.10 Errors and Error Handling

This section describes how the logical and physical layers detect RapidIO errors and respond to them. For details on the action of the SC3850 core when it is notified of any of these errors, see the *RapidIO Interconnect Specification, Revision 1.2,* part VII (Error Management Extensions Specifications).

16.2.10.1 RapidIO Error Description

RapidIO errors are classified under three categories: recoverable errors, notification errors, and fatal errors.

- *Recoverable errors*. Non-fatal transmission errors, such as a corrupt packet or corrupt control symbols and general protocol errors, for which there is hardware detection and recovery as described in the *RapidIO Interconnect Specification, Revision 1.2*. In these cases, the appropriate bit is set in the Port 0–1 Error Detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by the Port 0–1 Error Enable CSR) is captured in the Port 0–1 Error Capture CSRs. No interrupt is generated or actions required for a recoverable error. Recoverable errors are detected only in the physical layer.

- *Notification errors*. 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, a response to the interrupt is not crucial to port performance. 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 layers.
- *Fatal errors*. There are two types of fatal errors:
  - *Exceeded failed threshold*. The port fails because its recoverable error rate has exceeded a predefined failed threshold. RapidIO sets the Output Failed-encountered bit in the Port0 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).
  - *Exceeded consecutive retry*. The port fails because it has received too many packet retries in a row. The RapidIO controller sets the Retry Counter Threshold Trigger Exceeded bit in the Port 0–1 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 RapidIO controller internal queues and resume normal operation. Fatal errors are detected only in the physical layer.

16.2.10.2 Physical Layer RapidIO Errors

Table 16-8 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 can disable the error checking associated with a particular error. If this column is blank, error checking cannot be disabled. The Cause Field column indicates which 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 PnERECSR (see **page 16-165**) allows the error to increment the error rate counter and lock the Port 0–1 Error Capture registers—and also which PnEDCSR bit is set when the error is detected (see **page 16-164**).

Table 16-9 shows the RapidIO endpoint behavior after certain preset limits are exceeded (degraded threshold, failed threshold, retry threshold). **Table 16-10** shows the threshold response.

Table 16-8. 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 has 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 a packet with embedded idles.		Enter input error stopped.	5: Received invalid/illegal character		
1d	Received a control symbol with a bad CRC.	PnPCR[CCC] enables detect.	Enter input error stopped. Enter output error stopped.	2: Received a control symbol with bad CRC	Received corrupt control symbol	CCS

Table 16-8. Physical RapidIO Errors Detected (Continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable/ Detect
1d	Missing start: Packet data received without 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 no packet is received.		Enter input error stopped.	7/31: General error		
1e	Received a stomp control symbol when no packet is 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
2a	Received packet with a bad CRC value.	PnPCR[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 stops when Input Port Enable = 0.	Enter input error stopped.	3: Non-maintenance packet reception stops	Not Captured	
2d	Any packet received while Port Lockout bit is set	All packet reception stops when Port Lockout bit is set.	Enter input error stopped.	3: Non-maintenance packet reception stops	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

Table 16-8. Physical RapidIO Errors Detected (Continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable/ Detect
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
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.			

Table 16-9. Physical RapidIO Threshold Response

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt Clear*
Notification Errors					
Error rate counter exceeded the degraded threshold.	PnERTCSR[ERDTT] > 0 and any bit in PnERCSR enables detect and interrupt generation.	Generate interrupt. Continue to operate normally.	Degraded threshold	PnESCSR[ODE]	Write 1 to PnESCSR [ODE]
Fatal Errors					
Consecutive retry counter exceeded the retry counter threshold trigger	PRETCR[RET] > 0 enables detect and interrupt generation	Generate interrupt. Port is in priority order.	Consecutive retry threshold	PnIECSR[RETE]	Write 1 to PnIECSR [RETE]

Table 16-9. Physical RapidIO Threshold Response (Continued)

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt Clear*
Error rate counter exceeded the failed threshold.	PnERTCSR[ERFTT] > 0 and any bit in PnERCSR enables detect and interrupt generation.	Generate Interrupt. Port behavior depends on PnCCSR[SPF] and PnCCSR[DPE]. Port can continue transmitting packets or stop sending output packets, keeping or dropping them.	Failed threshold	PnESCSR[OFE]	Write 1 to PnESCSR [OFE]
Note: Information given here is minimal for clearing the interrupt. More detailed steps should be taken to find the cause of the interrupt, as described in the interrupt generation reference of the *RapidIO Interconnect Specification, Revision 1.2* Part VII (Error Management Extensions Specifications).					

16.2.10.3 Logical Layer RapidIO Errors

This section describes how the logical layer detects and responds to RapidIO errors. The action of the core processor when it is notified of these errors is minimally described. For details, see the interrupt generation reference in the *RapidIO Interconnect Specification, Revision 1.2,* part VII (Error Management Extensions Specifications).

Table 16-10 through **Table 16-23** list all the errors detected by the RapidIO endpoint logical layer and the actions taken. Error responses are sent as follows:

- When the RapidIO endpoint action includes sending an error response to the system or the RapidIO interconnect, an error response is sent only if the original transaction is a request requiring a response. Otherwise, no error response is sent.
- For multiple errors, a discard of a packet has a higher priority than an error response.
- For misaligned transactions, the error management extension registers are updated with each child.

Table 16-10. Hardware Errors For NREAD Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Priority of read transaction is 3.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error valid is when passthrough is disabled and accept_all is disabled or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
DestID DestID does not match this port 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 | accept_all) is false.	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	

Table 16-10. Hardware Errors For NREAD Transaction (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
SourceID Not Checked for error.				
TransactionType Received RapidIO packet with reserved TType for this ftype.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	
RdSize Not checked for error.				
SrcTID Not checked for error.				
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-44.	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	
Address:WdPtr:Xambs Request hits a protected ATMU window or the local configuration space window.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	
Address:WdPtr:Xambs Beginning address matches LCSBA1CSR with no 32-bit read request. Performed only when ttype == 4'b0100.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	
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 0s. Padding of 0s is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	
PayloadSize Not Applicable.				

The Logical/Transport Layer Address Capture Command and Status Register described on **page 16-161** uses the incoming RapidIO packet for a small transport packet as follows:

- LTLACCSR[XA] gets packet bits 78–79.
- LTLACCSR[A] gets packet bits 48–76.
- LTLDIDCCSR[DIDMSB] gets 0s.
- LTLDIDCCSR[DID] gets packet bits 16–23.
- LTLDIDCCSR[SIDMSB] gets 0s.
- LTLDIDCCSR[SID] gets packet bits 24–31.
- LTLCCCSR[FT] gets packet bits 12–15.
- LTLCCCSR[TT] gets packet bits 32–35.
- LTLCCCSR[MI] gets 0s.

The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows:

- LTLACCSR[XA] gets packet bits 94–95.
- LTLACCSR[A] gets packet bits 64–92.
- LTLDIDCCSR[DIDMSB] gets packet bits 16–23.
- LTLDIDCCSR[DID] gets packet bits 24–31.
- LTLDIDCCSR[SIDMSB] gets bits 32–39.
- LTLDIDCCSR[SID] gets packet bits 40–47.
- LTLCCCSR[FT] gets packet bits 12–15.
- LTLCCCSR[TT] gets packet bits 48–51.
- LTLCCCSR[MI] gets 0s.

Table 16-11. Hardware Errors For Maintenance READ/WRITE Request Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Priority of maintenance read or write request transaction is 3.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is 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	RapidIO packet is dropped.
DestID DestID does not match this port 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 | accept_all) is false.	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	
SourceID Not Checked for error.				
TransactionType Reserved transaction type for this ftype	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes	RapidIO packet is dropped.
RdSize Read/Write request size is not for 4 bytes.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes	
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 a small transport packet or not 16 bytes for large transport packet. Maintenance Write request Total header size is not 12 bytes for a small transport packet or not 16 bytes for a large transport packet. Padding of 0s in last two bytes of large transport packet is not checked.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes	
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	

Table 16-11. Hardware Errors For Maintenance READ/WRITE Request Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
The Logical/Transport Layer Address Capture Command and Status Register described on **page 16-161** uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-12. Hardware Errors For NWRITE, NWRITE_R

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not UT	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is 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	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
SourceID Not applicable.				
TransactionType	Yes if LTLEECSR[UT] is set.	LTLEDCSR[UT]	Yes	
TransactionType Received RapidIO packet with reserved TType for this ftype. Packet is treated as Nwrite Transaction.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE	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.	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.	RapidIO packet is dropped for NWRITE.
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window or the local configuration space window.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
Address:WdPtr:Xambs Write request hits overlapping ATMU windows. Refer to **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-44.	Yes if LTLEECSR[IACB] is set.	LTLEDCSR[IACB]	Yes for NWRITE_R. No for NWRITE.	RapidIO packet is dropped for NWRITE.
SrcTID Not Checked for error.				
Address:WdPtr:Xambs NWRITE_R address matches LCSBA1CSR with a request that is not a 32-bit read. Performed only for NWRITE_R packet.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	Yes for NWRITE_R.	

Table 16-12. Hardware Errors For NWRITE, NWRITE_R (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
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 is present. Large transport packet has 14 valid bytes and two 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.	RapidIO packet is dropped for NWRITE.
PayloadSize Not unsupported transaction. Payload is greater than that indicated by the {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.	RapidIO packet is dropped for NWRITE.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-13. Hardware Errors For SWRITE Transactions

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Swrite transaction priority is 3.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is 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	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped.
SourceID Not checked for error.				
Address:WdPtr:Xambs SWRITE request hits overlapping ATMU windows. See **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-44.	Yes if LTLEECSR[IACB] is set.	LTLEDCSR[IACB]	No	RapidIO packet is dropped.
Address:WdPtr:Xambs Request hits a protected ATMU window or the local configuration space window.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.

Table 16-13. Hardware Errors For SWRITE Transactions (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
PayloadSize Payload size is not in DWs, has exceeded 256 bytes, or has no payload.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 62–63. • LTLACCSR[A] gets packet bits 32–60. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 46–76. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-14. Hardware Errors For Maintenance Response Transactions

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not unsupported response. Response priority is not higher than the RapidIO maintenance priority.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped and ignored.
Received TT that is not enabled. Error is 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	RapidIO packet is dropped and ignored.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped and ignored.
SourceID Does not match the request DestID.	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
Transaction Type. Not unsupported transaction. Received RapidIO packet with reserved TType for the FType.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
Not unsupported response. Maintenance read/write response does not correspond to an outstanding valid message read/write request.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
HopCount Not checked for error.	—	—	—	—
Status Not unsupported response. Is not “Done” or “Error” Not “Done” status for “read_response” transaction type with payload “Error” status with payload.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
Status Not unsupported response Error Response.	Yes if LTLEECSR[IER] is set.	LTLEDCSR[IER]	Yes	OCN error response is generated to requestor
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
Header Size Not unsupported response 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 packet or is less than 16 bytes for large transport packet. Padding of 0s for small or large transport packet is not verified.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[TID]	No	RapidIO packet is dropped and ignored.

Table 16-14. Hardware Errors For Maintenance Response Transactions (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
PayloadSize Not unsupported response Maintenance writeresponse—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 64-bit aligned.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	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	OCN response is generated to requestor.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets packet bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[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 0s.				

Table 16-15. Hardware Errors For IO Response Transactions (Not Maintenance)

Error	Interrupt Generated	Status Bit Set	OCN 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 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	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
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

Table 16-15. Hardware Errors For IO Response Transactions (Not Maintenance) (Continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Not UR IO read response does not correspond to an outstanding valid IO read request. IO write response does not correspond to an outstanding valid IO 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.	OCN 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.	OCN 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.

Table 16-15. Hardware Errors For IO Response Transactions (Not Maintenance) (Continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
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.
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.	OCN error response is generated to requestor.

Table 16-15. Hardware Errors For IO Response Transactions (Not Maintenance) (Continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
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.	OCN 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 OCN done response is generated when the Done_With_Data is received for non misaligned requests or the last child of a misaligned 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.

Table 16-15. Hardware Errors For IO Response Transactions (Not Maintenance) (Continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Not UR Response for OCN 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 OCN 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 16-16. Hardware Errors For Message Request Transactions

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not applicable.				
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is 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	RapidIO packet is dropped.
DestID DestID does not match this port 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 | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped.	If priority is 3, packet is dropped.
SourceID Not checked for error.				
MsgLen,Ssize,Ltr,Mbox,MsgSeg Not checked for error.				
PayloadSize Message payload size is larger than the specified ssize or is of size 0 when seg_len == msg_len, or message payload size is not equal to specified ssize when seg_len!= msg_len.	Yes if LTLEECSR[MFE] is set.	LTLEDCSR[MFE]	Yes if priority is not 3. Else packet is dropped.	If priority is 3, packet is dropped.
Reserved ssize field.	Yes if LTLEECSR[MFE] is set.	LTLEDCSR[MFE]	Yes if priority is not 3. Else packet is dropped.	If priority is 3, packet is dropped.

Table 16-16. Hardware Errors For Message Request Transactions (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
Other Received message request with SOCAR[M] disabled.	Yes if LTLEECSR[UT] is set.	LTLEDCSR[UT]	Yes if priority is not 3. Else packet is dropped.	If priority is 3, packet is dropped.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets packet bits 56–63.				

Table 16-17. Hardware Errors For Message Response Transactions

Error	Interrupt	Status Bit Set	OCN Error Response	Comments
Priority Not checked for error.				
TransportType Receive reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped and ignored.
Received TT that is not enabled. Error is 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	RapidIO packet is dropped and ignored.
DestID (All non-maintenance) DestID does not match this port 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 | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped and ignored.
SourceID Not Checked for error.				
Status Not checked for error.				
Other Received message response with SOCAR[M] disabled.	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets packet bits 56–63. For all entries except the first, the capture registers are loaded from the response RapidIO packet.				

Table 16-18. Hardware Errors For Doorbell Request Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not applicable.				
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No.	RapidIO packet is dropped.
Received TT that is not enabled. Error is 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.	RapidIO packet is dropped.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	Yes if 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.	Yes if LTLEECSR[UT] is set.	LTLEDCSR[UT]	Yes if priority is not 3. Else packet is dropped.	

The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows:

- LTLACCSR[XA] gets packet bits 78–79.
- LTLACCSR[A] gets packet bits 48–76.
- LTLDIDCCSR[DIDMSB] gets 0s.
- LTLDIDCCSR[DID] gets packet bits 16–23.
- LTLDIDCCSR[SIDMSB] gets 0s.
- LTLDIDCCSR[SID] gets bits 24–31.
- LTLCCCSR[FT] gets packet bits 12–15.
- LTLCCCSR[TT] gets packet bits 32–35.
- LTLCCCSR[MI] gets 0s.

The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows for all entries but the blank ones:

- LTLACCSR[XA] gets packet bits 94–95.
- LTLACCSR[A] gets packet bits 64–92.
- LTLDIDCCSR[DIDMSB] gets packet bits 16–23.
- LTLDIDCCSR[DID] gets packet bits 24–31.
- LTLDIDCCSR[SIDMSB] gets packet bits 32–39.
- LTLDIDCCSR[SID] gets packet bits 40–47.
- LTLCCCSR[FT] gets packet bits 12–15.
- LTLCCCSR[TT] gets packet bits 48–51.
- LTLCCCSR[MI] gets 0s.

Table 16-19. Hardware Errors For Doorbell Response Transactions

Error	Interrupt	Status Bit Set	OCN Error Response	Comments
Priority Not UR or UT Response priority is not higher than RapidIO request priority.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped and ignored.
Received TT Error is 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	RapidIO packet is dropped and ignored.
DestID DestID does not match this port DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error is valid when (passthrough | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped and ignored.
SourceID Does not match the request's DestID.	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
Status Not UR or UT Not one of Done/Error/Retry.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
TransactionType Not UR or UT Anything other than Done_No_Data.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.
TargetTID No outstanding transaction for this TargetTID.	Yes if LTLEECSR[UR] is set.	LTLEDCSR[UR]	No	RapidIO packet is dropped and ignored.
Packet Size Not UR or UT DMA response. Header size is not 8 bytes for small transport packet or not 12 bytes for large transport packet. Two byte padding of 0s in a large transport field packet is not checked.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped and ignored.

Table 16-19. Hardware Errors For Doorbell Response Transactions

Error	Interrupt	Status Bit Set	OCN Error Response	Comments
Packet response time-out Response is not received by configured time.	Yes if LTLEECSR[PRT] is set.	LTLEDCSR[PRT]	Yes	
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows for all entries but the last one, in which the capture registers are loaded from original request RapidIO packet: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bit 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-20. Hardware Errors for Port-Write Transaction

Error	Interrupt	Status Bit Set	Error Response	Comments
Priority Not applicable.				
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	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	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 | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped.
SourceID Not checked for error.				
TransactionType Not checked for error.				
WrSize Not UT Is one of reserved sizes or less than 4 bytes.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
SrcTID Not checked for error.				
HopCount Not checked for error.				
ConfigOffset Not checked for error.				
PayloadSize Not UT An incorrect port-write wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56, or 64 bytes). Payload size is greater than the value defined by wr_size. Payload size is not 64-bit aligned when the wr_size is not 4 bytes.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.

Table 16-20. Hardware Errors for Port-Write Transaction (Continued)

Error	Interrupt	Status Bit Set	Error Response	Comments
Other Received PortWrite transaction with DOCAR[PW] disabled.	Yes if LTLEECSR[UT] is set.	LTLEDCSR[UT]	No	RapidIO packet is dropped.
In **Table 16-20**, the Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. In **Table 16-20**, the Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows for all entries but the blank ones: • LTLACCSR[XA] gets packet bits 94-95 • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bit 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-21. Hardware Errors for Outbound Transaction Crossed ATMU Boundary

Error	Interrupt	Status Bit Set	OCN Error Response Generated	Comments
OCN Address and payload size OCN address range for Outbound RapidIO transaction hits multiple ATMU windows.	Yes if LTLEECSR[IACB] is set and/or LTLEDCSR[PRT] is set	LTLEDCSR[OACB] LTLEDCSR[PRT]	Yes if original request requires a response.	OCN error response is generated if the request requires a response. Otherwise, the OCN packet is dropped.
The Logical/Transport Layer Address Capture Command and Status Register uses the original RapidIO packet sent out by outbound. For a small transport packet, it uses the following: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. For a large transport packet, it uses the following: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-22. Hardware Errors for Outbound Packet Time-to-Live Errors

Error	Interrupt	Status Bit Set	OCN Error Response Generated	Comments
Packet time-to-live error.	Yes if LTLEECSR[PTTL] is set	LTLEDCSR[PTTL]	Yes if original request requires a response.	OCN error response is generated if the request requires a response. Otherwise, the OCN packet is dropped.
The Logical/Transport Layer Address Capture Command and Status Register uses the RapidIO packet attempted to send outbound. For a small transport packet, it uses the following: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. For a large transport packet, it uses the following: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s.				

Table 16-23. Hardware Errors for Reserved Ftype

Error	Interrupt Generated	Status Bit Set if Corresponding Bit is Enabled	Error Response	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 | accept_all is enabled | transaction is addressed to this port).	Yes if LTLEECSR[UT] is set.	LTLEDCSR[UT]	No	RapidIO packet is dropped.
TransportType Received reserved TT.	Yes if LTLEECSR[TSE] is set.	LTLEDCSR[TSE]	No	RapidIO packet is dropped.
Received TT that is not enabled. Error is 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	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 | accept_all) is false.	Yes if LTLEECSR[ITTE] is set.	LTLEDCSR[ITTE]	No	RapidIO packet is dropped.
Address:WdPtr:Xambs Swrite request hits overlapping ATMU windows. Refer to **Section 16.2.5.4.2**, *Window Boundary Crossing Errors,* on page 16-44. Packet is checked as a non-SWRITE packet.	Yes if LTLEECSR[IACB] is set.	LTLEDCSR[IACB]	No	RapidIO packet is dropped.
Address:WdPtr:Xambs Not UT Request hits a protected ATMU window or the local configuration space window. Packet is checked as non-Swrite packet.	Yes if LTLEECSR[ITD] is set.	LTLEDCSR[ITD]	No	RapidIO packet is dropped.
The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a small transport packet as follows: • LTLACCSR[XA] gets packet bits 78–79. • LTLACCSR[A] gets packet bits 48–76. • LTLDIDCCSR[DIDMSB] gets 0s. • LTLDIDCCSR[DID] gets packet bits 16–23. • LTLDIDCCSR[SIDMSB] gets 0s. • LTLDIDCCSR[SID] gets bits 24–31. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 32–35. • LTLCCCSR[MI] gets 0s. The Logical/Transport Layer Address Capture Command and Status Register uses the incoming RapidIO packet for a large transport packet as follows for all entries: • LTLACCSR[XA] gets packet bits 94–95. • LTLACCSR[A] gets packet bits 64–92. • LTLDIDCCSR[DIDMSB] gets packet bits 16–23. • LTLDIDCCSR[DID] gets packet bits 24–31. • LTLDIDCCSR[SIDMSB] gets packet bits 32–39. • LTLDIDCCSR[SID] gets packet bits 40–47. • LTLCCCSR[FT] gets packet bits 12–15. • LTLCCCSR[TT] gets packet bits 48–51. • LTLCCCSR[MI] gets 0s. • LTLCCCSR[MI] gets 0s.				

16.3 RapidIO Message Unit

The RapidIO message unit handles multicast and non-multicast single-segment messages and non-multicast multiple-segment messages. The RapidIO controller has two inbound and two outbound message controllers.

The message passing programming model for inter-processor and inter-device communication enables a producer to send a message across the interconnect fabric to the message hardware of a consumer, called a mailbox. The receiving mailbox hardware places the message into a queue located in local memory. A message consists of one to sixteen segments. When a configured number of messages is received, an interrupt is generated (if enabled) to the interrupt controller for the processor to process the messages. Messages can be queued for transmission in the producer memory, and the message hardware processes them sequentially. Messages can also be queued in consumer memory while software processes them sequentially. Software can configure the depths of the queues in the producer and consumer memories. A multicast function allows single-segment messages to be sent to multiple consumers.

The message unit is compliant with the message passing logical specification in the *RapidIO Interconnect Specification, Revision 1.2*. The message passing model is most commonly used in systems in which a processing element is allowed to access only memory that is local to itself, processing elements communicate with each other through message passing, and communication is address independent.

Each inbound message controller has a dedicated interrupt to notify software that a configured number of messages have been received. Similarly, each outbound message controller has a dedicated interrupt to notify software that there are no more messages for the outbound mailbox controller. The message controller is managed through a set of run-time registers.

16.3.1 Features

The message unit supports two outbound message controllers with the following features:

- Chaining and direct modes.
- Multicast up to 32 RapidIO destinations for single-segment messages.
- Transmission to any mailbox for a single-segment message.
- Transmission to any mailbox for a multi-segment message.
- Segment size up to 256 bytes.
- Up to 16 segment messages with a total payload of up to 4 KB.
- One entire message with up to a 16 message segments can be transmitted before a response is received.
- One entire single-segment message to all multicast destinations (up to 32) can be transmitted before a response is received.

- All message segment transfers for a message transaction must complete before the next message transaction begins.
- Pipelined transmission of a full message in each message controller but all responses must be received before the next message can be transmitted.
- In Chaining mode, the next descriptor can be fetched before the current message completes (descriptor prefetching).

The message unit supports two inbound message controllers with the following features:

- Reception of any mailbox and letter for a single- or multi-segment message.
- Segment size up to 256 bytes.
- Up to sixteen segment messages with a total payload of up to 4 KB.
- Full inbound line rate performance.
- Back-to-back message reception of the same or lower priority.
- Out-of-order message segment reception.
- Concurrent inbound message controller operation.

16.3.2 Outbound Message Controller Operation

The outbound message controller sends messages stored in local memory, and it can operate in three different modes:

- *Direct mode*. Software programs the necessary registers to point to the beginning of the message in memory.
- *Chaining mode*. Software programs the necessary registers to point to the beginning of the first valid descriptor in memory. The descriptor provides all the necessary registers to start the message transfer.
- *Multicast mode*. A single-segment message can be sent to multiple destinations. Multicast mode is supported in Direct or Chaining mode.

Each outbound message controller uses a unique identifying number. For example, if there are two outbound message controllers, message controller 0 uses number 0 and message controller 1 uses number 1.

16.3.2.1 Direct Mode

In Direct mode (OM*x*MR[MUTM] = 1; see **page 16-194**) the outbound message controller does not read descriptors from memory. Instead, it uses the parameters programmed in the outbound message controller registers to start the transfer. Software initializes all the parameters to start the message transmission. The message transfer starts when the outbound message controller start bit, OM*x*MR[MUS] changes from 0 to 1 and the outbound message controller is not busy. If it is busy, OM*x*MR[MUS] the transition from 0 to 1 is ignored. Software should program all the appropriate registers before setting OM*x*MR[MUS].

There are many ways in which software can interact with the message controller. One example sequence of events to start and complete a transfer in Direct mode is as follows:

1. Poll the OM*x*SR[MUB] bit to ensure that the outbound message controller is not busy.
2. Clear the following OMxSR status bits (see **Table 16-104, *OMxSR Field Descriptions,*** on page 16-197):
   - MER
   - RETE
   - PRT]
   - TE
   - QOI
   - QFI
   - EOMI
   - QEI
3. Initialize the following registers:
   - Source address (OM*x*SAR)
   - Destination port (OM*x*DPR)
   - Destination attributes (OM*x*DATR)
   - Retry error threshold (OM*x*RETCR)
   - Double-word count (OM*x*DCR).

   If multicast mode is enabled (OM*x*MR[MM]), initialize the multicast group and list in OM*x*MGR and OM*x*MLR.
4. Initialize the outbound message mode register message unit transfer mode bit, OM*x*MR[MUTM] = 1, to indicate direct mode. Other control parameters must also be initialized in the mode register.
5. Clear and then set the mode register message unit start bit, OM*x*MR[MUS], to start the message transfer.
6. The outbound message controller sets the OM*x*SR[MUB] bit to indicate that the message transfer is in progress.
7. The outbound message controller reads a message segment from local memory using the source address register (OM*x*SAR).
8. If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
9. After the message read to local memory completes, the message is sent.

10. The outbound message controller clears OM*x*SR[MUB]. A non-multicast message transfer completes after all message segments complete. A multicast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
    - Done response received
    - Error response
    - Packet response time-out received
    - Retry error threshold exceeded
    - An internal error occurs during the local memory access
11. After the outbound message operation completes, the outbound message interrupt is generated if the end of message outbound message interrupt event is enabled (OMnDATR[EOMIE]).

    In Direct mode, the outbound message interrupt is generated after a message operation completes if OM*x*DATR[EOMIE] =1.

    The event causing this interrupt is indicated by OM*x*SR[EOMI]. The interrupt is held until the OM*x*SR[EOMI] bit is cleared by writing a 1 to it.

    In Direct mode, the Error/Port-Write interrupt is generated for the following reasons:
    - A message error response is received and this interrupt event is enabled (OM*x*MR[EIE]).
    - A packet response time-out occurs and this interrupt event is enabled (OM*x*MR[EIE]).
    - A retry threshold exceeded error occurs and this interrupt event is enabled (OM*x*MR[EIE]).
    - An internal error response is received and this interrupt event is enabled (OM*x*MR[EIE]).

Table 16-24 describes each of these error types.

Table 16-24. Error Types In Outbound Message Controller Direct Mode

Error Type	Message Controller Response to Error
Message Error Response	• Sets the message error response status bit (OM*x*SR[MER]). • Generates a serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).

Table 16-24. Error Types In Outbound Message Controller Direct Mode (Continued)

Error Type	Message Controller Response to Error
Packet Response Time-Out	• Sets the packet response time-out status bit (OM*x*SR[PRT]). • Generates a serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Retry Error Threshold Exceeded	• Sets the retry threshold exceed status bit (OM*x*SR[RETE]). • Generates a Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Internal Error During Local Memory Read	• Sets the transaction error bit (OM*x*SR[TE]) • Does not send message segments with an internal error because the message data is not available • Does not transfer memory reads generated before the internal error. • Generates but does not transfer additional memory reads for the same message operation. • Does not transfer all subsequent message segments for the same message operation, including retried message segments. • Stops after the message operation completes (indicated by OM*x*SR[MUB]). • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set.

16.3.2.2 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the message controller has stopped operation by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding OMxSR status bit (see **Table 16-104, *OMxSR Field Descriptions,*** on page 16-197):
   - MER
   - PRT
   - RETE
   - TE

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Determines that an error has occurred by polling the status bits OMxSR status bit (see **Table 16-104, *OMxSR Field Descriptions,*** on page 16-197):
   - MER

- PRT
- RETE
- TE

2. Verifies that the message controller has stopped operation by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding status bit (listed in step 1).

16.3.2.3 Disabling and Enabling the Message Controller

Once the message controller is started, it cannot be stopped except by loss of power or reset.

16.3.2.4 Hardware Error Handling

Table 16-25 describes Direct mode hardware errors. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers. These error condition checks are provided by the messaging unit in addition to the error condition checks provided by the RapidIO port described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-48.

Table 16-25. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message request	Internal error during a read of the message segment from local memory	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if OM*x*MR[EIE] set **Status bit set**: Transaction error in the outbound message status register (OM*x*SR[TE]). Message failed in the mailbox CSR (MCSR[FA]). **Message segment sent:** No **Logical/Transport Layer Capture Register:** **Comments:** Message controller stops after the current message operation completes. The descriptor dequeue pointer is not incremented in chaining mode.
Message request	Internal error for an earlier message segment local memory read. An internal error for a subsequent message segment local memory read for the same message may or may not occur.	**Error checking level:** 2 **Interrupt generated:** **Status bit set**: None **Message segment sent:** Yes **Logical/Transport Layer Capture Register:**
Undefined packet	Reserved ftype encoding[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT] **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.

Table 16-25. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message response	Reserved tt encoding[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Message response	Illegal destination ID[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	ttype (transaction field) is not message response[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Message response received and no outbound mailboxes are supported[1]	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]. **Message segment sent:** No **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Reserved response status (not done, retry, or error)	**Error checking level:** 4a **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Message response packet size is incorrect	**Error checking level:** 4a **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.

Table 16-25. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message response	Incorrect source ID	**Error checking level:** 4b **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Letter, mbox and msgseg not outstanding or letter, mbox not outstanding	**Error checking level:** 4b **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set. **Status bit set**: Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	RapidIO priority is less than or equal to message request	**Error checking level:** 4c **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments:** Packet is ignored and discarded.
Message response	Error response	**Error checking level:** 5 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MER] set. Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. OM*x*SR[MER] bit is set in Direct mode or Chaining mode. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding message request packet.[2] **Comments:** Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.
Message response	Number of retries exceeds limit	**Error checking level:** 5 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[RETE] set. Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Retry error threshold exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE]. OM*x*SR[RETE] bit is set in Direct mode or Chaining mode. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding message request packet.[2] **Comments:** Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.

Table 16-25. Outbound Message Direct Mode Hardware Errors

Transaction	Error	Description
Message response	Packet response time-out	**Error checking level:** Unrelated **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[PRT] set. Serial RapidIO error/write-port if OM*x*MR[EIE]. **Status bit set**: Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT]. OM*x*SR[PRT] bit is set in Direct mode or Chaining mode. **Message segment sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding message request packet.[2] The LTLDIDCCSR[SIDMSB] and LTLDIDCCSR[SID] field has a value of 0. **Comments:** Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.
Notes: 1.	**Notes: 1.** These error types are actually detected in the RapidIO port, not in the message controller. **2.** In small transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 78–79). • LTLACCSR[A] gets the address (packet bits 48–76). • LTLDIDCCSR[MDID] gets 0. • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[MSID] gets 0. • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 32–35). • LTLCCCSR[MI] gets the msg info (packet bits 40–47) In large transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 94–95). • LTLACCSR[A] gets the address (packet bits 64–92). • LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31). • LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39). • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 48–51) if the message request packet is captured or 0 if the message response packet is captured. • LTLCCCSR[MI] gets the message information (packet bits 56–63).

Table 16-26 lists programming errors that result in undefined or undesired hardware operation.

Table 16-26. Outbound Message Direct Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Double word count greater than 256 bytes when multi-cast mode selected	No	None	Undefined operation
Double word count set to a reserved value	No	None	Undefined operation
Transaction flow level set to 3	No	None	Undefined operation
Target interface set to an invalid RapidIO port	No	None	Undefined operation
Source address for message read is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Register values changed during operation	No	No	Undefined operation.

16.3.2.5 Chaining Mode

In Chaining mode, OM*x*MR[MUTM] = 0, message descriptors are built in local memory in a circular queue. Several options are available to the programmer. Software can build one or more descriptors before initializing the outbound message controller registers, or it can initialize the outbound message controller registers and then build the descriptors. Software maintains the enqueue pointer (OM*x*DQEPAR). The outbound message controller dequeues descriptors, processes them, and increments the dequeue pointer (OM*x*DQDPAR) to point to the next descriptor in the queue. **Figure 16-9** depicts a sample structure of the outbound portion of the message controller and a descriptor queue, with each valid descriptor queue entry pointing to a valid message. The descriptor queue has eight entries, four of which are valid. The local processor enqueues descriptors and the outbound message controller dequeues the descriptors.

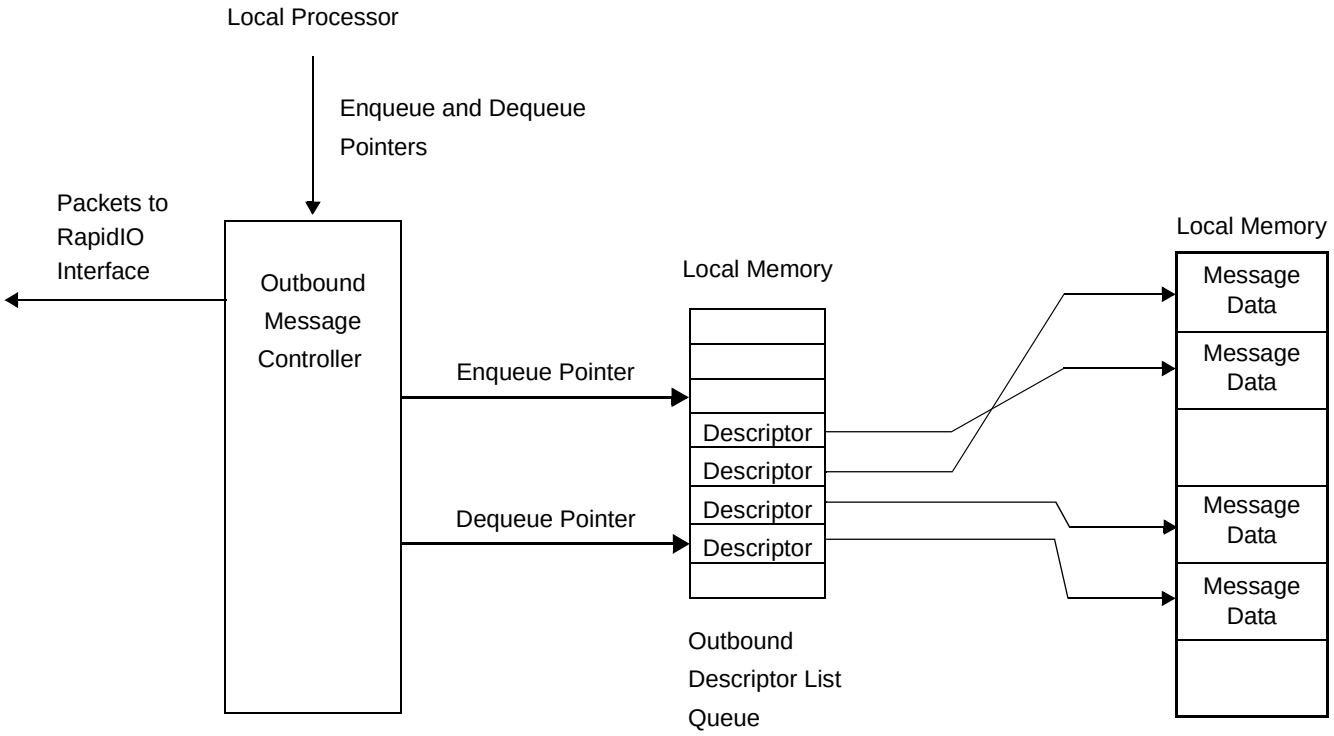


Figure 16-9. Outbound Frame Queue Structure

One of several ways software can initialize the message controller in Chaining mode is as follows:

1. Poll the status register message unit busy bit, OM*x*SR[MUB], to verify that the outbound message controller is not busy.
2. Clear the message unit start bit (OM*x*MR[MUS]).
3. Initialize the descriptor queue dequeue pointer address registers (OM*x*DQDPAR; see **page 16-198**) and the enqueue pointer address registers (OM*x*DQEPAR; see **page 16-203**) to the same value for proper operation.

These registers must also be queue size aligned on a boundary equal to the number of queue entries × 32 bytes (the size of each queue descriptor). For example, there are 16 entries in the queue, the register must be 512-byte aligned.

The number of queue entries is set in OMnMR[CIRQ_SIZ]. See **Section 16.6.59**, *Outbound Message x Mode Registers (OMxMR)*, on page 16-194.

4. Initialize the retry error threshold in the outbound message retry error threshold configuration register (OM*x*RETCR; see **page 16-204**).
5. Clear OM*x*MR[MUTM] for Chaining mode.
6. If you are using single-segment multicast mode, set OM*x*MR[MM].
7. Configure the other control parameters in the mode register (OM*x*MR).
8. Clear OM*x*SR[MER, PRT, RETE, TE, QOI, QFI, EOMI, and QEI]. If OM*x*SR[MER, PRT, RETE, TE, or QOI] are not cleared, the message controller does not start a new message operation. Incorrect status is indicated if the other status bits are not cleared.
9. Set the message unit start (OM*x*MR[MUS]) to enable the outbound message controller and cause the descriptor queue dequeue pointer (OM*x*DQDPAR) to be saved as the base address of the descriptor queue.

The method to start and complete transfers by adding descriptors after initializing the message unit is as follows:

1. Create one or more descriptors in local memory starting at the address to which the descriptor queue enqueue pointer address register (OM*x*DQEPAR) is pointing.
2. Either increment the enqueue pointer address registers (OM*x*DQEPAR) by setting OM*x*MR[MUI] for each descriptor entry added or directly change the enqueue pointer address register (OM*x*DQEPAR). If software sets OM*x*MR[MUI], the message controller clears this bit after successfully incrementing the enqueue pointer.
3. When the descriptor queue is not empty, the message controller reads the descriptor from local memory using the address to which the dequeue pointer (OM*x*DQDPAR) is pointing and sets the busy bit (OM*x*SR[MUB]).
4. The message controller reads the next descriptor from local memory, if available, using the address to which the dequeue pointer (OM*x*DQDPAR) is pointing. The message controller does not prefetch more than one descriptor.
5. The message controller sets OM*x*SR[MUB] to indicate that the message transfer is in progress. OM*x*SR[MUB] remains set until the descriptor queue is empty or a transaction error occurs.

6. Software can create and enqueue additional descriptors while the message controller is busy (OM*x*SR[MUB]). Software can continue adding descriptors as long as the descriptor queue is not full. If software adds descriptors using the OM*x*MR[MUI] bit, overflowing the queue can be prevented by polling the queue full bit (OM*x*SR[QF]) before creating and enqueueing the next descriptor.
7. After the descriptor memory read completes, the corresponding message segment is read from local memory.
8. If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
9. After the message read to local memory completes, the message is sent.
10. If multi-cast is enabled, all the indicated targets are sent the same message.
11. A non-multicast message transfer completes after all message segments complete. A multi-cast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
    - Done response received
    - Error response received
    - Packet response time-out
    - Retry error threshold exceeded
    - Internal error during the descriptor (all message segments complete) or message read of local memory
12. When processing for the current descriptor completes and another descriptor is available, the preceding steps are repeated.
13. If a RapidIO error response is received, the message error response bit is set (OM*x*SR[MER]) and the outbound message controller operation stops after all message segments complete. If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.
14. If a packet response time-out occurs, the packet response time-out bit is set (OM*x*SR[PRT]). If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.
15. If the retry error threshold value is exceeded for a specific segment, the retry error threshold exceeded bit is set (OM*x*SR[RETE]) and outbound message controller operation stops after all message segments complete. If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.
16. If an internal error occurs while local memory is read, the transaction error bit is set (OM*x*SR[TE]) and outbound message controller operation stops after all message segments complete. If OM*x*MR[EIE] is set, the interrupt Serial RapidIO error/write-port is generated.

This process continues until the descriptor queue is empty (dequeue pointer equals the enqueue pointer).

17. The message unit clears OM*x*SR[MUB] after it processes the last descriptor or a transaction error occurs.
18. If an error occurs, the message unit must be disabled, reinitialized, and reenabled before another message can be sent.

16.3.2.5.1 Changing Descriptor Queues in Chaining Mode

When software switches to another descriptor queue in local memory, it must wait for the processing of the current queue to complete, as indicated by the busy bit (OM*x*SR[MUB]). Software then disables the message controller by clearing OM*x*MR[MUS], changes the enqueue and dequeue descriptor pointers (OM*x*DQEPAR and OM*x*DQDPAR), and reenables the message unit by setting OM*x*MR[MUS].

16.3.2.5.2 Preventing Queue Overflow in Chaining Mode

Software must guarantee that descriptors are not added to an already full queue. When the increment bit is used (OM*x*MR[MUI]), software can poll the queue full bit (OM*x*SR[QF]) before enqueueing another descriptor. When software sets the enqueue pointer directly, software is responsible for not overflowing the descriptor queue.

16.3.2.5.3 Switching Between Direct and Chaining Modes

The message unit architecture allows switching from Direct mode to Chaining mode and *vice versa* after all required parameters are initialized in the appropriate registers and the message unit is not busy, as indicated by clearing of the OM*x*SR[MUB]. If OM*x*MR[MUS] is cleared and then set during a switch from Direct mode to Chaining mode, the message unit is reinitialized in Chaining mode and the outbound message descriptor dequeue pointer address is saved as the new base address of the circular queue in memory. During a switch from Chaining mode to Direct mode, OM*x*MR[MUS] must also be cleared and set.

16.3.2.5.4 Chaining Mode Descriptor Format

The descriptor contains information the message unit controller needs to transfer data. Software must ensure that each descriptor is aligned on a 32-byte boundary and that the descriptor queue is on a queue boundary, that is, on a boundary equal to the number of queue entries × 32 byte (the size of each queue descriptor). For each descriptor in the queue, the message unit controller starts a new message operation with the control parameters specified by the descriptor, as shown in **Table 16-27**.

Table 16-27. Outbound Message Unit Descriptor Summary

Descriptor Field	Description
Reserved	—
Source address	Source address of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Source Address Register.
Destination port	Destination port of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Destination Port Register.
Destination attributes	Transaction attributes of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Destination Attributes Register.
Multicast group	Logical multi-cast group. Groups are defined as a list of 32 numerically consecutive destinations by deviceID.
Multi-cast list	Bit vector list of consecutive destinations by deviceID.
Double-word count	Number of double-words for the message operation. After the message controller reads the descriptor from memory, this field is loaded into the Double-Word Count Register.
Reserved	—

Figure 16-10 depicts the queue dequeue pointer and an associated descriptor. The descriptor is valid only if the enqueue and dequeue pointers are not equal.

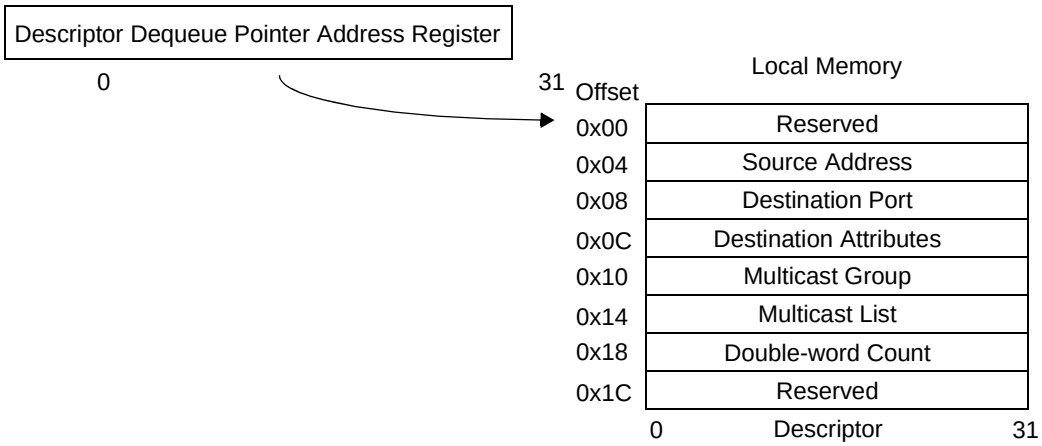


Figure 16-10. Descriptor Dequeue Pointer and Descriptor

16.3.2.5.5 Chaining Mode Controller Interrupts

An outbound message interrupt can be generated for one of the following reasons.

- *Queue Empty*. The queue goes empty and the interrupt event is enabled (OM*x*MR[QEIE] = 1). The event causing the outbound message interrupt is indicated by OM*x*SR[QEI]. The interrupt is held until the queue is not empty and the OM*x*SR[QEI] bit is cleared by writing a value of 1 to it.
- *Queue Full*. The queue is full and the interrupt event is enabled (OM*x*MR[QFIE] = 1). The event causing the outbound message interrupt is indicated by OM*x*SR[QFI]. The interrupt is held until the queue is not full and the OM*x*SR[QFI] bit is cleared by writing a value of 1 to it.
- *Queue Overflow*. The queue is full, the increment bit is set (OM*n*MR[MUI]), and the interrupt event is enabled (OM*x*MR[QOIE] = 1). The event causing the outbound message interrupt is indicated by OM*x*SR[QOI]. The message unit must be reinitialized. The interrupt is held until the OM*x*SR[QOI] bit his cleared by writing a value of 1 to it. This interrupt is also generated if the enqueue pointer is directly written and causes an overflow.
- *End-Of-Message*. The message completed and the interrupt event is enabled (OM*x*DATR[EOMIE] = 1). The event causing this interrupt is indicated by OM*x*SR[EOMI]. The interrupt is held until the OM*x*SR[EOMI] bit is cleared by writing a value of 1 to it. This allows an interrupt to be generated after a particular descriptor is processed.

An error/port-write interrupt can be generated for the following reasons in Chaining mode.

- *Message error response*. Message error response is received and this interrupt event is enabled (OM*x*MR[EIE])
- *Packet response time-out.* A packet response time-out occurs and this interrupt event is enabled (OM*x*MR[EIE]).
- *Retry error threshold exceeded*. A retry threshold exceeded error occurs and this interrupt event is enabled (OM*x*MR[EIE]).
- *Transaction error*. A message error response is received and this interrupt event is enabled (OM*x*MR[EIE]).

Table 16-24 describes each of these error types.

Table 16-28. Error Types In Outbound Message Controller Direct Mode

Error Type	Message Controller Response to Error
Message Error Response	• Sets the message error response status bit (OM*x*SR[MER]). • Generates the Serial RapidIO error/write-port if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Packet Response Time-Out	• Sets the packet response time-out status bit (OM*x*SR[PRT]). • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Retry Error Threshold Exceeded	• Sets the retry threshold exceed status bit (OM*x*SR[RETE]) • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set. • Stops after the message operation completes (indicated by OM*x*SR[MUB]).
Transaction error	• Sets the transaction error bit (OM*x*SR[TE]). • Does not generate message segment reads and sends no message segments if the internal error occurs while the memory controller reads descriptor memory. • Does not sent message segments with an internal error because the message data is not available. • Does not transfer memory reads generated before the internal error. • Generates additional memory reads for the same message operation but does not transfer them. • Does not transfer subsequent message segments for the same message operation. This includes retried message segments. • Stops after the message operation completes (indicated by OM*x*SR[MUB]). • Generates the Serial RapidIO error/write-port interrupt if OM*x*MR[EIE] is set.

16.3.2.6 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the message controller has stopped operation by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding OM*x*SR status bit (MER, PRT, RETE, or TE).

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Determines that an error has occurred by polling the following OM*x*SR status bits (see **Table 16-104, *OMxSR Field Descriptions,*** on page 16-197:
   - MER
   - PRT
   - RETE
   - TE
2. Verifies that the message controller has stopped by polling OM*x*SR[MUB].
3. Disables the message controller by clearing OM*x*MR[MUS].
4. Clears the error by writing a 1 to the corresponding OM*x*SR status bit (listed in step 1).

16.3.2.7 Hardware Error Handling

Table 16-29 shows error conditions in addition to those for Direct mode. All the errors listed in **Table 16-25, *Outbound Message Direct Mode Hardware Errors,*** on page 16-84 can also occur. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers. The messaging unit provides these error condition checks in addition to the error condition checks provided by the RapidIO port and described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-48.

Table 16-29. Additional Hardware Error Conditions

Transaction	Error	Description
Message request	Internal error during a read of the descriptor from local memory	**Error checking level:** 0 **Interrupt generated:** Serial RapidIO error/write-port if OM*x*MR[EIE] is set. **Status bit set**: Transaction error in the outbound message status register (OM*x*SR[TE]). Message Failed in the Mailbox CSR (MCSR[FA]). **Message segment sent:** No **Logical/Transport Layer Capture Register:** **Comments:** Message controller stops. Note that the descriptor dequeue pointer is not incremented.

Table 16-30 lists programming errors that result in undefined or undesired hardware operation. These errors are in addition to those listed in **Table 16-26, *Outbound Message Direct Mode Programming Errors,*** on page 16-87.

Table 16-30. Outbound Message Chaining Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Enqueued descriptor address is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Address for descriptor enqueue address pointer is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Descriptor enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation
Descriptor queue size set to a reserved value	No	No	Undefined operation
Address of descriptor enqueue pointer set to a value outside of queue	No	No	Undefined operation
Enqueueing of descriptors causes descriptor queue overflow	Outbound message interrupt enable set (OM*x*MR[QOIE])	Queue overflow (OM*x*SR[QOI])	Message controller stops.
Queue misaligned	No	No	May result in duplicate messages being sent.

16.3.2.8 Outbound Message Controller Arbitration

There are two ways to define the order in which each message controller sends messages from its message queues when both outbound message controllers are enabled:

- *Fixed priority*. The lowest numbered message unit has the highest priority. Message unit 0 has the highest priority.
- *Rotating priority*. The message units take turns sending messages in round-robin (message units 0, 1, 0, 1 and so on). A message controller can send 1–64 messages.

OM*x*MR[SCNTL] (see **page 16-194**) configures the message units for the fixed or rotating modes of arbitration.

In fixed-priority arbitration, all message segments for message unit 0 must complete before message unit 1 can start. However, in rotating priority arbitration, the other message unit can start processing a message as soon as all message segments are transmitted. Also, in fixed-priority arbitration when a message unit other than message unit 0 is processing a message, message unit 0 can start processing a message as soon as all message segments are transmitted.

16.3.3 Inbound Message Controller Operation

The inbound message controller receives messages and places them in a circular frame queue in local memory. It can receive segments of a message in any order. The address to which a message is to be written is computed as: Base address + (msgseg × ssize in double-words). In contrast to the outbound message controller for which software controls the enqueue pointer and hardware controls the dequeue pointer, the inbound message controller controls the enqueue pointer and the software controls the dequeue pointer.

Figure 16-11 depicts a sample structure of the inbound message frames and the frame pointers. In this example, the frame queue has eight entries, three of which are currently valid. The inbound controller controls the enqueue pointer and the software controls the dequeue pointer. After a configured number of messages is received, an interrupt is generated to the processor. After processing a received message, the local processor can either write the inbound message mode register mailbox increment bit (IM*x*MR[MI]]), causing the dequeue pointer to point to the next message frame in the queue, or wait until all received messages are processed and write to the dequeue pointer.

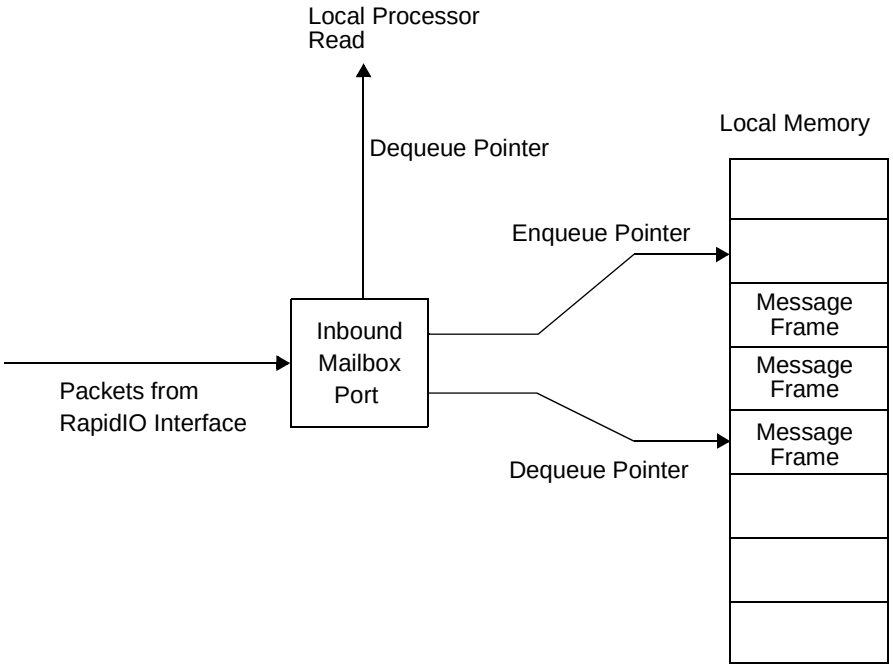


Figure 16-11. Inbound Message Structure

16.3.3.1 Inbound Message Controller Initialization

The sequence of events to initialize the inbound message controller is as follows:

1. Initialize the frame queue dequeue pointer address registers (IM*x*FQDPAR) and the frame queue enqueue pointer address registers (IM*x*FQEPAR) to the same value for proper operation. These registers must also be queue size aligned, that is, they must be aligned on a boundary equal to the number of queue entries × frame size in byte. For example, if there are eight entries in the queue and the frame size is 128 bytes, the register must be 1024-byte aligned.
2. Clear the status register (IM*x*SR).
3. In the inbound message mode register (IM*x*MR), set the mailbox enable bit (IM*x*MR[ME]) along with the other control parameters (frame queue size, message-in-queue threshold, frame size, snoop enable, and the various interrupt enables).

16.3.3.2 Inbound Controller Operation

There are many ways in which software can interact with the message controller. One method is as follows:

1. The inbound message controller receives a message segment request from the RapidIO port. If the inbound message controller is enabled (IM*x*MR[ME] =1), the inbound message controller has received all the segments for the previous message, and the frame queue is not full then the message segment is accepted.
2. The inbound message controller computes the address for each segment of the message (up to 16 segments per message) using the value of the inbound message frame queue enqueue pointer address registers and the segment number.
3. The inbound message controller writes each segment to the circular queue in local memory at the computed address.
4. When the entire message is received and all message segments are written, the inbound message controller increments the enqueue pointer to point to the next message frame in local memory. A message operation completes after all message segments for the message are complete. A message segment is complete when one of the following occurs:
    - The memory write completes (either successfully or with an internal error).
    - A message request time-out occurs (all message segments not yet received are now complete).

5. The message segments of one message can immediately be followed by the message segments of another message under the following conditions:
   - If a message segment of a new message arrives before all memory writes complete for the previous message and the RapidIO priority (PRIO field value in the message packet) of the new message is equal to or lower than that of all previous messages, the message controller processes the message and a memory write is generated to the appropriate frame queue entry.
   - If the RapidIO priority of the new message is higher than that of any previous message memory writes not yet complete, the message controller generates a retry.
   - If a message segment arrives before all previous message memory writes for the same message complete and the new message segment has a higher priority than the previous message segments, the message controller generates a retry.

   The message controller clears the IM*x*SR[MB] bit when all message operations complete.
6. An inbound message interrupt is generated to the local processor if the number of messages in the queue is greater than or equal to the configured message-in-queue threshold (IM*x*MR[MIQ_THRESH) and this event is enabled to generate the interrupt (IM*x*MR[MIQIE]).
7. By reading the inbound message status register (IM*x*SR[MIQI]), software detects that the message-in-queue interrupt bit is set and determines that the message-in-queue event caused the interrupt.
8. Software processes the frame queue entry to which the frame dequeue pointer address registers (IM*x*FQDPAR) are pointing.
9. Software increments the dequeue pointer address registers (IM*x*FQDPAR) by setting the message increment bit (IM*x*MR[MI]). Software determines whether there are any more messages to process by reading the queue empty bit (IM*x*SR[QE]). If the queue is not empty, the previous two steps are repeated.
10. Optionally, software reads the enqueue pointer address registers (IM*x*FQEPAR) and processes all the received messages. After message processing is complete, the dequeue pointer address registers (IM*x*FQDPAR) are written.
11. Software clears the message-in-queue interrupt bit (IM*x*SR[MIQI]) by writing a 1 to the IM*x*SR[MIQI] bit.

16.3.3.3 Message Steering

Messages are forwarded to the inbound message controllers as follows:

- Messages directed to mailbox 0 are forwarded to message controller 0.
- Messages directed to mailbox 1, 2, or 3 are forwarded to message controller 1.

16.3.3.4 Retry Response Conditions

The conditions to generate a logical layer retry (response retry) are as follows:

- The local memory circular queue is full and a message is received.
- The inbound message controller has received at least one segment of a multiple-segment message but has not received all message segments (as determined by a different RapidIO source ID, RapidIO destination ID, or mailbox).
- A message is received with a higher priority than all previous messages being written to memory, but it has not completed.

Note: If all inbound messages have the same RapidIO priority, this condition for generating a retry does not occur.

16.3.3.5 Inbound Message Controller Interrupts

The inbound message controller generates the inbound message interrupt for several different events. Each event can be individually enabled:

- Message-In-Queue interrupt is generated under one of the following conditions:
  - The circular queue has accumulated the specified number of messages, and this interrupt event is enabled (IM*x*MR[MIQIE]). The event causing this interrupt is indicated by IM*x*SR[MIQI]. The interrupt is held until the dequeue and enqueue pointers indicate that the specified number of messages is not in the frame queue and the IM*x*SR[MIQI] bit is cleared by writing a 1 to it.
  - The circular queue contains one or more messages, the specified number of messages has not accumulated, a message has not been dequeued for the maximum interrupt report interval, and this interrupt event is enabled (IM*x*SR[MIQIE]). The event causing this interrupt is indicated by IM*x*SR[MIQI]. The interrupt is held until the IM*x*SR[MIQI] bit is cleared by writing a 1 to it.
- Queue Full interrupt is generated when the circular queue becomes full and this interrupt event is enabled (IM*x*SR[QFIE]). The event causing this interrupt is indicated by IM*x*SR[QFI]. The interrupt is held until the queue is not full and the IM*x*SR[QFI] bit is cleared by writing a 1 to it.

The error/port-write interrupt is generated for the following reasons.

- An interrupt is generated after a message request time-out and this interrupt event is enabled (IM*x*MR[EIE]). The message request time-out counter starts after the first valid segment of a multi-segment message is received and the time-out counter is enabled.
- A transaction error interrupt is generated after an internal error response is received and this interrupt event is enabled (IM*x*MR[EIE]).

Table 16-31 describes each of these error types.

Table 16-31. Error Types In the Inbound Message Controller

Error Type	Message Controller Response to Error
Message Request Time-Out	• Sets the message request time-out status bit (IM*x*SR[MRT]). • Treats as complete all message segments not yet received. • Generates the Serial RapidIO error/write-port interrupt if IM*x*MR[EIE] is set. • Stops after all message segments complete (indicated by IM*x*SR[MB]).
Transaction Error	• Sets the transaction error bit (IM*x*SR[TE]) and enters the error state. • Returns an error response. Memory writes already generated before the internal error also return an error response. • Stops after the message operation completes (indicated by IM*x*SR[MB]). • Generates the Serial RapidIO error/write-port interrupt if IM*x*MR[EIE] is set.

16.3.3.6 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error
2. Verifies that the message controller has stopped operation by polling IM*x*SR[MB].
3. Clears the error by writing a 1 to the corresponding status bit (IM*x*SR[MRT] and/or IM*x*SR[TE]).
4. Disables, reinitializes, and reenables the message unit before another message can be received.
5. Reinitializes and re-enables the message controller.

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions (see **Table 16-115, *IMxSR Field Descriptions,*** on page 16-209):

1. Determines that an error has occurred by polling the status bits (IM*x*SR[MRT] and/or IM*x*SR[TE]).
2. Verifies that the message controller has stopped operation by polling IM*x*SR[MB].
3. Disables the message controller by clearing IM*x*MR[ME].
4. Clears the error by writing a 1 to the corresponding status bit (IM*x*SR[MRT] and/or IM*x*SR[TE])

16.3.3.7 Hardware Error Handling

Table 16-32 lists the hardware error conditions. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. After an error is detected, no additional error checking beyond the current level is performed. Note that messages are processed in a pipeline. The first error detected in the processing pipeline updates the error management extensions registers. These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port and described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-48.

Table 16-32. Inbound Message Hardware Errors

Error	Description
Reserved ftype[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded. An error or illegal transaction target error response is not generated.

Table 16-32. Inbound Message Hardware Errors

Error	Description
Illegal destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Incorrect message packet size[1] Payload is not the specified ssize except for the last segment. The last segment payload can be less than or equal to the segment size but not 0. **Note:** Payload sizes are 64-bit multiples.	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. Comments: Packet is ignored and discarded.
Reserved ssize field[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet **Comments**: Packet is ignored and discarded.
Message received and no mailboxes are supported as indicated by DOCAR[M][1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Message packet size larger than the configured frame size and message controller is enabled	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.
Inbound message packet with a RapidIO priority of 3	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue entry written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments**: Packet is ignored and discarded.

Table 16-32. Inbound Message Hardware Errors

Error	Description
Not an error. The RapidIO priority is not consistent for all message segments of a message	**Error checking level:** 2 **Interrupt generated:** **Status bit set**: **Queue entry written in local memory:** Yes **Response status:** Done or retry **Logical/Transport Layer Capture Register:** **Comments:** Retry response occurs if the higher-priority message segment is received while the memory write for a corresponding lower-priority message segment is outstanding.
Message segment number is larger than the number of message segments in the message	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
Duplicate message segment is received All segments of a multi- segment message must be received before the next message begins.	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
msglen (number of segments in a message) is not consistent in all segments of a multi-segment message	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No. **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
ssize (segment size) is not consistent in all segments of a multi-segment message	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded.
Message received for an unsupported mailbox but at least one mailbox is supported	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with the packet. **Comments:** Packet is ignored and discarded. This error applies only if a mailbox is not supported. This error is not currently supported since all mailboxes are supported.

Table 16-32. Inbound Message Hardware Errors

Error	Description
Message controller disabled and message received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Message controller enabled but in the error state and message received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Internal error during the write of the frame queue entry to memory	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if IM*x*MR[EIE] is set. **Status bit set**: Transaction error in the Message Status Register (IM*x*SR[TE]). Message failed in the Message CSR (MCSR[FA]). **Queue entry written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:** Message controller stops after the current message operation completes. The enqueue pointer is not incremented.
Internal error for an earlier posted frame queue entry memory write and a subsequent frame queue entry memory write is posted before the internal error is detected. An internal error may or may not occur during the subsequent frame queue entry memory write. The frame queue could be for the same message or a different message	**Error checking level:** 4 **Interrupt generated:** No **Status bit set**: None **Queue entry written in local memory:** Yes **Response status:** Error **Logical/Transport Layer Capture Register:** **Comments:**

Table 16-32. Inbound Message Hardware Errors

Error	Description
Message request to request time-out for multi-segment messages	**Error checking level:** Unrelated **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MRT] set. Serial RapidIO error/write-port if IMxMR[EIE] is set. **Status bit set**: Message request time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MRT]. IM*x*SR[MRT] bit is set. **Queue entry written in local memory:** No **Response status:** No **Logical/Transport Layer Capture Register:** Updated with the previous message request packet except that the message segment field (bits 4–7 of LTLCCCSR[MI]) is updated with the lowest message segment number not yet received.[2] **Comments:** All message segments received before the time-out update memory. The enqueue pointer is not incremented. The message operation completes.

Notes:

1. These error types are actually detected in the RapidIO port, not in the message controller.

2. In small transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 78–79).
- LTLACCSR[A] gets the address (packet bits 48–76).
- LTLDIDCCSR[MDID] gets 0.
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23).
- LTLDIDCCSR[MSID] gets 0.
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31).
- LTLCCCSR[FT] gets the ftype (packet bits 12–15).
- LTLCCCSR[TT] gets the ttype (packet bits 32–35).
- LTLCCCSR[MI] gets the msg info (packet bits 40–47).

In large transport size configuration using the packet, the following allocations are made:

- LTLACCSR[XA] gets the extended address (packet bits 94–95)
- LTLACCSR[A] gets the address (packet bits 64–92)
- LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23)
- LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31)
- LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39)
- LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47)
- LTLCCCSR[FT] gets the ftype (packet bits 12–15)
- LTLCCCSR[TT] gets the ttype (packet bits 48–51)
- LTLCCCSR[MI] gets the msg info (packet bits 56–63).

16.3.3.8 Programming Errors

Table 16-33 shows a partial list of programming errors that result in undefined or undesired hardware operation.

Table 16-33. Inbound Message Programming Errors

Error	Interrupt	Status Bit Set	Comments
Reserved value of the message in queue threshold (IMxMR[MIQ_THRESH]) or reserved value of the circular frame queue size (IMxMR[CIRQ_SIZE])	No	No	Undefined operation results
The message in-queue threshold is equal to the frame queue size	No	No	Message in queue interrupt occurs when queue is full
The message in-queue threshold is greater than the frame queue size	No	No	Message in queue interrupt never occurs
Frame queue entry written to non-existent memory	No	No	Memory controller will cause the interrupt and update capture registers. IM*x*SR[TE] will be set due to the internal error.
Message enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
The dequeue frame pointer register is set incorrectly.	No	No	Undefined operation results
Queue misaligned.	No	No	Undefined operation results.

16.3.3.9 Disabling and Enabling the Inbound Message Controller

When the message controller is disabled by clearing IM*x*MR[ME] the following occurs (see **Table 16-115, *IMxSR Field Descriptions,*** on page 16-209):

1. Queue full clears the IMxSR[QF] bit.
2. Message-in-queue clears the IMxSR[MIQ] bit.
3. Queue empty is set via the IMxSR[QE] bit.
4. Message busy clears the IMxSR[MB] bit after all pending frame queue entry writes complete.

Once the message controller is disabled, an error response is generated for all new message packets. If the message controller is disabled before all of the message segments for a multisegment message are received, a message request time-out must occurs and all pending frame queue writes must complete before message busy clears (IMxSR[MB]).

Before the message controller is reenabled, the message busy bit must be clear (IM*x*SR[MB = 0) and the (IM*x*MR[ME]), frame queue dequeue pointer address registers (IM*x*FQDPAR), and frame queue enqueue pointer address registers (IM*x*FQEPAR) must be initialized to the same value for proper message controller operation.

16.3.4 RapidIO Message Passing Logical Specification Register Bits

The Mailbox Command and Status Register (MCSR; see **page 16-139**) provides the status for the two inbound and the two outbound controllers. These read-only status bits indicate the state of each of the message controllers, as described in **Table 16-34**.

Table 16-34. MCSR Bits to Indicate Status of Inbound and Outbound Controllers

MCSR Bit	Description
Available (A)	Indicates the following: • The inbound message controller is enabled (IM*x*MR[ME]). • The inbound message controller is not in the internal error state (IM*x*SR[TE] = 0). • The inbound message controller did not detect a message request time-out (IM*x*SR[MRT] = 0).
Full (FU)	Reflects the inbound message controller queue full status.
Empty (EM)	Reflects the state of the outbound message controller message empty status.
Busy (B)	Reflects the state of the inbound message controller busy bit IM*x*SR[MB].
Failed (FA)	Set if any of the following bits are set: • Inbound message controller transaction error status bit IM*x*SR[TE]. • Inbound message controller message request time-out status bit IM*x*SR[MRT]. • Outbound message controller transaction error status bit OM*x*SR[TE]. • Outbound message controller packet response time-out bit OM*x*SR[PRT]. • Outbound message controller message error response received status bit OM*x*SR[MER]. • Outbound message controller retry error threshold exceeded status bit OM*x*SR[RETE].
Error (ERR)	Always has a value of 0.

16.4 RapidIO Doorbell

This section describes the operation of the doorbell controllers, which are part of the RapidIO message unit. The doorbell controllers are designed to comply with the message passing logical specification contained in the *RapidIO Interconnect Specification*, Revision 1.2. The doorbell controller generates and receives doorbells.

The doorbell controllers are controlled through a set of run-time registers.

16.4.1 Features

- Support for one outbound doorbell controllers
- Support for one inbound doorbell controllers
- The doorbell controller can sustain back-to-back inbound doorbells

16.4.2 Doorbell Controller

The RapidIO architecture specification defines a doorbell type with no data payload that is transferred using inbound and outbound doorbell controllers. The MSC8156E RapidIO interconnect doorbell unit supports both inbound and outbound doorbells. The doorbell entry size is fixed at 64 bits because doorbell packets only pass a small amount of information, making the enqueue and dequeue pointers double-word addresses.

Inbound doorbells are handled by the doorbell controller similar to the way the message controller handles inbound data messages. The doorbell controller receives the doorbells and places them in a circular queue located in local memory. Additional doorbells can be received and forwarded to memory before the previous doorbell memory write completes as long as the RapidIO priority is the same or lower than the previous doorbell. The doorbell is retried if the RapidIO priority is higher than the previous doorbell.

Table 16-12 depicts an example of the structure of the inbound doorbell queue and pointers. The doorbell queue has eight entries, three of which are currently valid. The doorbell controller enqueues doorbells and the local processor dequeues doorbells.Each inbound doorbell controller has a dedicated interrupt to notify software that there are incoming doorbells.

Outbound doorbells are generated through a memory-mapped write port rather than a queue. An outbound doorbell must complete before another outbound doorbell can be generated. The outbound doorbell controller has a dedicated interrupt used to notify software that a doorbell was sent.

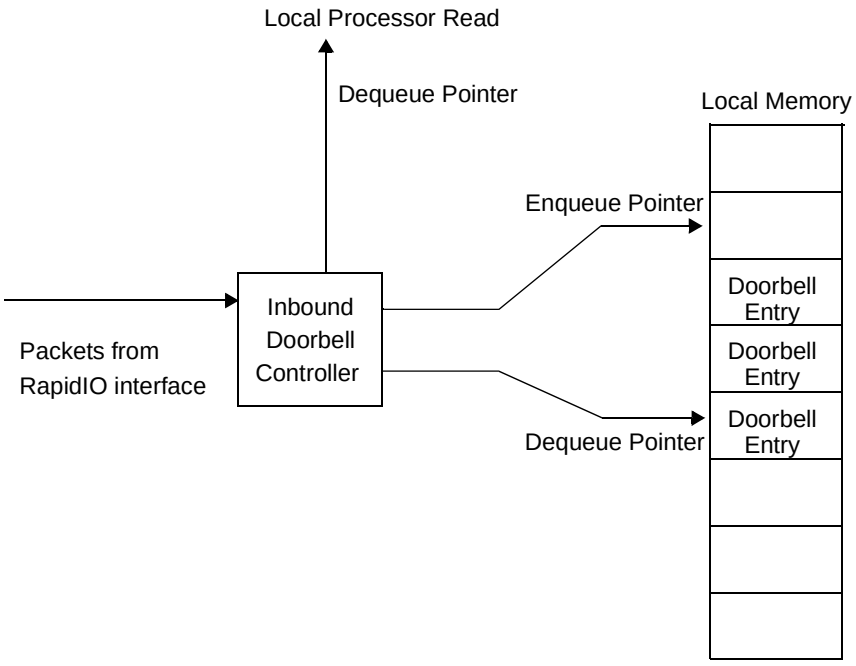


Figure 16-12. Inbound Doorbell Queue and Pointer Structure

16.4.3 Outbound Doorbell Controller

The outbound doorbell controller generates doorbells. Software initializes all parameters in the necessary registers to start the doorbell transmission. The doorbell transfer starts when the doorbell start bit, DUS, in the Outbound Doorbell Mode Register (ODMR) transitions from 0 to 1 (see **Table 16-119, *ODMR Field Descriptions,*** on page 16-214) and the doorbell controller is not busy. Software should program all appropriate registers before setting ODMR[DUS].

Software can interact with the doorbell controller in many ways. One example method for generating a doorbell is as follows:

1. Poll the status register doorbell unit busy bit, ODSR[DUB], to ensure that the outbox is not busy (see **Table 16-120, *ODSR Field Descriptions,*** on page 16-215).
2. Clear the following ODSR status bits:
   - MER
   - RETE
   - PRT
   - EODI]
3. Initialize the following registers:
   - Destination port (ODDPR; see **page 16-216**)
   - Destination attributes (ODDATR; see **page 16-217**)
   - Retry error threshold (ODRETCR; see **page 16-218**)
4. To start the doorbell transfer, clear and then set the doorbell start bit, ODMR[DUS].
5. ODSR[DUB] is set when ODMR[DUS] transitions from 0 to 1 to indicate that the doorbell transfer is in progress.
6. The outbound doorbell controller sends the doorbell.
7. The outbound doorbell controller clears the ODSR[DUB] bit after the doorbell operation completes. A doorbell completes when one of the following events occurs:
   - Done response received.
   - Error response received.
   - Packet response time-out.
   - Retry limit exceeded.
8. The outbound doorbell interrupt is generated if the end of doorbell outbound doorbell interrupt event is enabled (ODDATR[EODIE]).

16.4.3.1 Interrupts

The outbound doorbell controller interrupt is generated after the completion of a doorbell (done, error, packet response time-out, or retry limit exceeded) if this interrupt event is enabled (ODDATR[EODIE] =1). The event causing this interrupt is indicated by ODSR[EODI]. The interrupt is held until the ODSR[EODI] bit is cleared by writing a 1 to it.

Table 16-35 describes each of these error types.

Table 16-35. Error Types In the Outbound Doorbell Controller

Error Type	Doorbell Controller Response to Error
Error Response Error	• Sets the message error response status bit (ODSR[MER]). • Stops after the doorbell operation completes (indicated by ODSR[DUB]).
Packet Response Time-Out Error	• Sets the packet response time-out status bit (ODSR[PRT]). • Stops after the doorbell operation completes (indicated by ODSR[DUB]).
Retry Error Threshold Exceeded Error	• Sets the retry threshold exceed status bit (ODSR[RETE]). • Stops after the doorbell operation completes (indicated by ODSR[DUB]).

16.4.3.2 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the doorbell controller has stopped operation by polling ODSR[DUB].
3. Disables the doorbell controller by clearing ODMR[DUS].
4. Clears the error by writing a 1 to the corresponding ODSR status bit (see **Table 16-120, *ODSR Field Descriptions,*** on page 16-215):
   - MER
   - PRT
   - RETE

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Determines that an error has occurred by polling the ODSR status bits (MER, PRT, and/or RETE).
2. Verifies that the doorbell controller has stopped operation by polling ODSR[DUB].
3. Disables the doorbell controller by clearing ODMR[DUS].
4. Clears the error by writing a 1 to the corresponding ODSR status bit (listed in step 1).

Note: Once the doorbell controller starts, it cannot be stopped.

16.4.3.3 Hardware Error Handling

Table 16-36 lists possible error conditions for outbound doorbells and how they are handled. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. Note that outbound doorbell responses are processed in a pipeline. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port as discussed in **Section 16.2.10**, *Errors and Error Handling*, on page 16-48.

Table 16-36. Outbound Doorbell Hardware Errors

Transaction	Error	Description
Undefined packet	Reserved ftype encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE] (see **page 16-159**). **Doorbell sent:** Yes. **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Large transport size in small transport mode or small transport size in large transport mode.[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Doorbell response	Illegal Destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Doorbell not outstanding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UR] is set. **Status bit set**: Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.

Table 16-36. Outbound Doorbell Hardware Errors (Continued)

Transaction	Error	Description
Doorbell response	ttype (transaction field) is not doorbell response[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	RapidIO priority is less than or equal to outbound request[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set (see **page 16-160**). **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Incorrect Source ID[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] set (see **page 16-160**). **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Reserved response status[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set (see **page 16-160**). **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Doorbell response packet size is incorrect[1]	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set (see **page 16-160**). **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE] (see **page 16-159**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the packet.[2] **Comments**: Packet is ignored and discarded.
Doorbell response	Error response	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MER] is set (see **page 16-160**). **Status bit set**: Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER] (see **page 16-159**). ODSR[MER] bit set (see **page 16-214**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding doorbell request packet.[2] **Comments**: Doorbell transfer complete.

Table 16-36. Outbound Doorbell Hardware Errors (Continued)

Transaction	Error	Description
Doorbell response	Number of retries exceeds limit	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[RETE] is set (see **page 16-160**). **Status bit set**: Retry limit exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] (see **page 16-159**). ODSR[RETE] bit is set (see **page 16-215**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the corresponding doorbell request packet.[2] **Comments**: Doorbell transfer complete.
Doorbell response	Packet response time-out[1]	**Error checking level:** Unrelated **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[PRT] is set (see **page 16-160**). **Status bit set**: Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT] in RapidIO endpoint (see **page 16-159**). ODSR[PRT] bit is set (see **page 16-215**). **Doorbell sent:** Yes **Logical/Transport Layer Capture Register:** Updated with the doorbell request packet in RapidIO endpoint.[2] **Comments**: Doorbell transfer complete. Note that RapidIO endpoint sends a special priority 3 packet indicating a doorbell time-out.
Notes: 1.	**Notes: 1.** These error types are actually detected in the RapidIO port, not in the doorbell controller. **2.** In small transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 78–79). • LTLACCSR[A] gets the address (packet bits 48–76) • LTLDIDCCSR[MDID] gets 0. • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[MSID] gets 0. • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 32–35). • LTLCCCSR[MI] gets 0 In large transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 94–95). • LTLACCSR[A] gets the address (packet bits 64–92). • LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31). • LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39). • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 48–51). • LTLCCCSR[MI] gets 0.

16.4.3.4 Programming Errors

Table 16-37 lists programming errors that result in undefined or undesired hardware operation.

Table 16-37. Outbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Transaction flow level set to 3	No	None	Undefined operation.
Target interface set to an invalid RapidIO port	No	None	Undefined operation.
Register values changed during operation	No	No	Undefined operation.

16.4.4 Inbound Doorbell Controller

The inbound doorbell controller receives doorbells and places them in a circular doorbell queue in local memory. The inbound controller controls the enqueue pointer and software controls the dequeue pointer. After a configured number of doorbells are received, an interrupt is generated to the processor. After processing a received doorbell, the local processor can either write the doorbell mode register increment bit (IDMR[DI]]) causing the dequeue pointer to point to the next doorbell in the queue or wait until all the received doorbells are processed and write the dequeue pointer.

There are many ways in which software can interact with the doorbell controllers. One method of initializing the inbound doorbell controller follows:

1. Initialize the doorbell queue dequeue pointer address registers IDQDPAR (see **page 16-222**) and IDQEPAR (see **page 16-223**) and the doorbell queue enqueue pointer address register (DQEPAR). These registers must be initialized to the same value for proper operation. They also must be queue size aligned.
2. Clear the status register (IDSR; see **page 16-221**).
3. Set the doorbell enable bit IDMR[DE] along with the other control parameters (doorbell queue size, doorbell-in-queue threshold, and various interrupt enables) in the doorbell mode register (IDMR; see **Table 16-124, *IDMR Field Descriptions,*** on page 16-219).

Another method follows:

1. The doorbell controller receives a doorbell. If the inbound doorbell controller is enabled (IDMR[DE] =1) and the doorbell queue is not full, then the doorbell is accepted.
2. The doorbell controller stores the 16-bit information field and the RapidIO source and destination IDs in local memory using the value of the doorbell queue enqueue pointer address register (DQEPAR).
3. When the memory write completes, the enqueue pointer is incremented to point to the next doorbell queue entry in local memory.
4. If another doorbell arrives before all previous doorbell memory writes complete and the RapidIO priority of the doorbell is equal to or lower than the priority of all previous doorbells, the doorbell controller processes the doorbell and generates a memory write

to the appropriate doorbell queue entry. If the priority of the new doorbell is higher than that of the previous doorbell memory writes that have not completed, the doorbell controller generates a retry.

5. An inbound doorbell interrupt is generated to the local processor because the number of doorbells in the queue is greater than or equal to the configured doorbell-in-queue threshold (IDMR[DIQ_THRESH) and this event is enabled to generate the interrupt (IDMR[DIQIE]).
6. Software determines that the doorbell-in-queue event caused the interrupt by detecting that the doorbell-in-queue interrupt bit is set in the doorbell status register (IDSR[DIQI]).
7. Software processes the doorbell queue entry to which the doorbell queue dequeue pointer address register (DQDPAR) is pointing.
8. Software increments the dequeue pointer address register (DQDPAR) by setting the doorbell increment bit (IDMR[DI]).
9. Software determines whether there are more doorbells to process by reading the queue empty bit (IDSR[QE]). If the queue is not empty, the previous two steps are repeated.
10. Software clears the doorbell-in-queue interrupt bit (IDSR[DIQI]) by writing a 1 to it.

16.4.4.1 Doorbell Queue Entry Format

Each doorbell entry in the queue has two 32-bit words, one for target information (see **Table 16-38**) and one for source information (see **Table 16-39**). The target information is stored because a RapidIO port can be configured to accept packets from any destination. When there are multiple RapidIO ports on one device, each port can be configured with a different destination ID. For the MSC8156E, the target information is identical for all received doorbell entries.

Table 16-38. Inbound Doorbell Target Info Definition

Bit	Name	Description
31–16	—	Reserved.
15–8	ETID	Extended target ID in Large Transport mode. Reserved for Small Transport mode.
7–0	TID	Target ID field from the received doorbell packet.

Table 16-39. Source Info Definition

Bit	Name	Description
31–24	ESID	Extended source ID in Large Transport mode. Reserved fir Small Transport mode.
23–16	SID	Source ID field from the received doorbell packet.
15–8	INFO MSB	Most significant byte of the info field from the received doorbell packet.
7–0	INFO LSB	Least significant byte of the info field from the received doorbell packet.

Figure 16-13 depicts the doorbell queue entry fields and their related offsets.

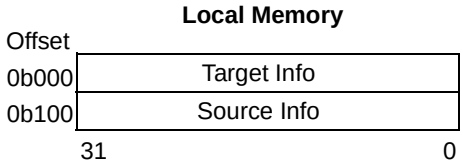


Figure 16-13. Doorbell Entry Format

16.4.4.2 Retry Response Conditions

There are two conditions in which a doorbell is retried at the logical layer (Response Retry):

- A doorbell is received and there are no entries in the doorbell queue.
- A doorbell is received with a higher priority than all previous doorbells being written to memory. If all inbound doorbells have the same priority, this condition does not occur.

16.4.4.3 Doorbell Controller Interrupts

There is one doorbell controller interrupt per inbound doorbell controller. The following events can generate the interrupt:

- *Doorbell-in-queue*:
  - The circular queue has accumulated the number of doorbells specified by the doorbell-in-queue threshold (IDMR[DIQ_THRESH) and this interrupt event is enabled (IDMR[DIQIE]). The event causing this interrupt is indicated by IDSR[DIQI]. The interrupt is held until the dequeue and enqueue pointers indicate that the specified number of doorbells is not in the doorbell queue and the IDSR[DIQI] bit is cleared by writing a 1 to it.
  - The circular queue contains one or more doorbells, the specified number of doorbells has not accumulated, a doorbell has not been dequeued for the maximum interrupt report interval, and this interrupt event is enabled (IDMR[DIQIE]). The event causing this interrupt is indicated by IDSR[DIQI]. The interrupt is held until either IDMR[DI] is set or DQDPAR[DQDPA] is written and IDSR[DIQI] is cleared by writing a 1 to it.
- *Queue Full*. An interrupt is generated each time the circular queue becomes full and this interrupt event is enabled (IDSR[QFIE]). The event causing this interrupt is indicated by IDSR[QFI]. The interrupt is held until the queue is not full and the IDSR[QFI] bit is cleared by writing a 1 to it.

The error/port-write interrupt can be generated after an internal error response is received and this interrupt event is enabled (IDMR[EIE]).

16.4.4.4 Transaction Errors

When an internal error occurs while the doorbell controller is writing to local memory the doorbell controller responds as follows:

1. Sets the transaction error bit (IDSR[TE]) and enters the error state.
2. Returns an error response.
3. Generates the Serial RapidIO error/write-port interrupt if IDMR[EIE] is set.

16.4.4.5 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software performs the following actions.

1. Determines the cause of the interrupt and processes the error.
2. Verifies that the doorbell controller has stopped operation by polling IDSR[DB].
3. Clears the error by writing a 1 to the corresponding status bit (IDSR[TE]).
4. Disables, reinitializes, and reenables the doorbell unit before another doorbell can be received.

16.4.4.6 Hardware Error Handling

Table 16-40 describes the hardware error conditions and how they are handled. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking beyond the current level is performed. Doorbells are processed in a pipeline. The first error detected in the processing pipeline updates the error management extensions registers. These error condition checks are provided by the messaging unit. These checks are in addition to the error condition checks provided by the RapidIO port as discussed in **Section 16.2.10**, *Errors and Error Handling*, on page 16-48.

Table 16-40. Inbound Doorbell Hardware Errors

Error	Description
Reserved ftype[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Large transport size in small transport size or small transport size in large transport size[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set:** Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]**.** **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal Destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Inbound doorbell received and inbound doorbells are not supported as indicated by DOCAR[D][1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Inbound doorbell packet with a RapidIO priority of 3	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] is set. **Status bit set**: Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.

Table 16-40. Inbound Doorbell Hardware Errors

Error	Description
Incorrect doorbell packet size (not one datum in small transport mode or not two datums in large transport mode)	**Error checking level:** 2 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[MFE] is set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Doorbell controller disabled and doorbell received	**Error checking level:** 3 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** Packet is ignored and discarded.
Doorbell controller enabled but in the error state and doorbell received	**Error checking level:** 3 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** Packet is ignored and discarded.
Internal error during the write of the doorbell queue entry to memory	**Error checking level:** 4 **Interrupt generated:** Serial RapidIO error/write-port if IDMR[EIE] is set. **Status bit set**: Transaction error in the Doorbell status register (IDSR[TE]). Doorbell Failed in the Port-write and Doorbell CSR (PWDCSR[FA]). **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** Doorbell controller stops after the current doorbell operation completes. The enqueue pointer is not incremented.

Table 16-40. Inbound Doorbell Hardware Errors

Error	Description
An internal error occurred for an earlier posted write of a doorbell queue entry to memory and a subsequent write of a doorbell queue entry to memory is posted before the internal error is detected.	**Error checking level:** 5 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** Yes **Response status:** Error **Logical/Transport Layer Capture Register** **Comments:** An internal error may or may not occur during the subsequent write of a doorbell queue entry to memory.

Notes:
1. These error types are actually detected in the RapidIO port, not in the doorbell controller.
2. In small transport size configuration using the packet, the following allocations are made:
   - LTLACCSR[XA] gets the extended address (packet bits 78–79).
   - LTLACCSR[A] gets the address (packet bits 48–76).
   - LTLDIDCCSR[MDID] gets 0.
   - LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23).
   - LTLDIDCCSR[MSID] gets 0.
   - LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31).
   - LTLCCCSR[FT] gets the ftype (packet bits 12–15).
   - LTLCCCSR[TT] gets the ttype (packet bits 32–35).
   - LTLCCCSR[MI] gets 0.

   In large transport size configuration using the packet, the following allocations are made:
   - LTLACCSR[XA] gets the extended address (packet bits 94–95).
   - LTLACCSR[A] gets the address (packet bits 64–92).
   - LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23).
   - LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31).
   - LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39).
   - LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47).
   - LTLCCCSR[FT] gets the ftype (packet bits 12–15).
   - LTLCCCSR[TT] gets the ttype (packet bits 48–51).
   - LTLCCCSR[MI] gets 0.

16.4.4.7 Programming Errors

Table 16-41 lists the programming errors that result in undefined or undesired hardware operation.

Table 16-41. Inbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the doorbell in queue threshold (IDxMR[DIQ_THRESH]) or reserved value of the circular doorbell queue size (IDxMR[CIRQ_SIZ]); see **Table 16-124, *IDMR Field Descriptions,*** on page 16-219.	No	No	Undefined operation results
The doorbell in-queue threshold is equal to the doorbell queue size.	No	No	Doorbell in queue interrupt when queue is full
The doorbell in-queue threshold is greater than the doorbell queue size.	No	No	Doorbell in queue interrupt never occurs

Table 16-41. Inbound Doorbell Programming Errors (Continued)

Error	Interrupt Generated	Status Bit Set	Comments
Doorbell queue entry written to non-existent memory.	No	No	Memory controller causes the interrupt and updates the capture registers
Doorbell enqueue and dequeue pointers are not initialized to the same value.	No	No	Undefined operation
The dequeue pointer register is set incorrectly.	No	No	Undefined operation

16.4.4.8 Disabling and Enabling the Doorbell Controller

When the doorbell controller is disabled by clearing IDMR[DE] the following occurs in the IDSR (see **Table 16-125, *IDSR Field Descriptions,*** on page 16-221):

1. Queue full clears (QF).
2. Doorbell-in-queue clears (DIQ).
3. Queue empty is set (QE)
4. Doorbell busy clears (DUB) after all pending doorbell queue entry writes to local memory complete.

Before the doorbell controller is reenabled, the doorbell busy bit must be clear (IDSR[DB) and the doorbell dequeue pointer address register (DQDPAR) and the doorbell queue enqueue pointer address register (DQEPAR) must be initialized to the same value for proper doorbell controller operation.

16.4.4.9 RapidIO Message Passing Logical Specification Registers

The port-write and doorbell command and status register (PWDCSR) includes several doorbell controller status bits. See **Section 16.6.9**, *Port Write and Doorbell Command and Status Register (PWDCSR)*, on page 16-141 for details.

These read-only status bits indicate the state of doorbell controller 0.

- *Available (PWDCSR[A])*. Indicates that the inbound doorbell controller is enabled (IDMR[DE]) and the doorbell controller is not in the internal error state (IDnSR[TE] = 0)
- *Full (PWDCSR[FU])*. This bit reflects the inbound doorbell controller queue full status bit (IDSR[QF])
- *Empty (PWDCSR[EM])*. This bit reflects the inverted state of the outbound doorbell busy bit (ODSR[DUB] = 0)
- *Busy (PWDCSR[B])*. This bit reflects the state of the inbound doorbell controller busy bit IDSR[DUB]

- *Failed (PWDCSR[FA])*. This bit reflects the state of the transaction error status bit IDSR[TE]
- *Error (PWDCSR[ERR])*. This bit is always a 0

16.5 Port-Write Controller

The implementation of the port-write controller is very similar to the inbound message and doorbell controllers except that only one queue entry is supported. The port write is an error reporting mechanism for a device that has no end-point to communicate with a control processor or other system host. **Figure 16-14** shows the structure of the inbound queue and pointer. The port-write queue only contains one entry with a fixed size of 64 bytes and is aligned to a cache line boundary. The port-write controller uses the error/port-write interrupt for the RapidIO error/write-port to indicate incoming port-writes.

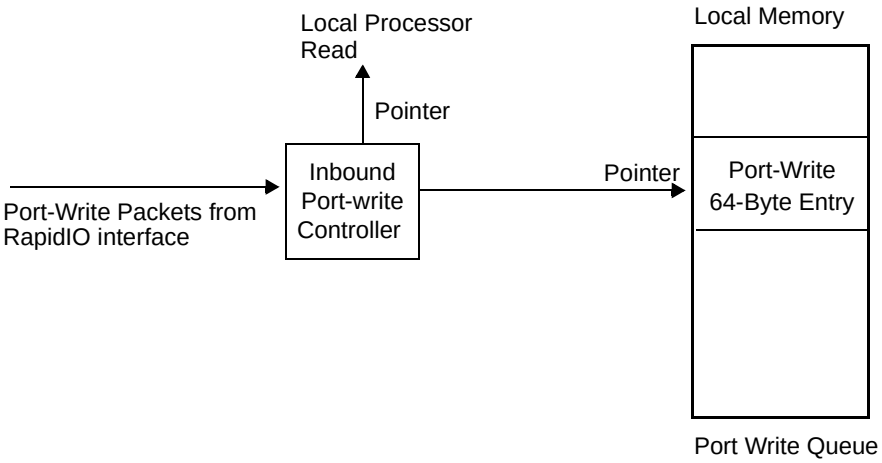


Figure 16-14. Inbound Port-Write Structure

16.5.1 Port-Write Controller Initialization

There are many ways in which software can interact with the port-write controller. One method to initialize the port-write controller is as follows:

1. Initialize the port-write queue base address registers (IPWQBAR; see **Table 16-131, *IPWQBAR Field Descriptions,*** on page 16-227).
2. Clear the status register (IPWSR; see **Table 16-130, *IPWSR Field Descriptions,*** on page 16-226).
3. Set the port-write enable bit IPWMR[PWE] along with the interrupt enable) in the inbound port-write mode register (IPWMR; see **Table 16-129, *IPWMR Field Descriptions,*** on page 16-225).

16.5.2 Port-Write Controller Operation

There are several ways in which software can interact with the port-write controller. One method is as follows:

1. The port-write controller receives a port-write from the RapidIO port. If the inbound port-write controller is enabled (IPWMR[PWE] =1) and the port-write queue is not full, the port-write is accepted.
2. The port-write controller stores 64 bytes of payload in local memory using the value of the port-write queue base address registers (IPWQBAR). Valid payload sizes include 4, 8, 16, 24, 32, 40, 48, 56, or 64 bytes. Note that 64 bytes are always written to memory. If the actual payload size is less than 64 bytes, the non payload data is undefined.
3. If the queue full interrupt enable bit is set (IPWMR[QFIE]) after the memory write completes, the port-write controller generates the error/port-write interrupt.
4. An inbound error/port-write interrupt is generated to the local processor because a port-write is received and this event is enabled to generate the interrupt (IPWMR[QFIE]). Note that the RMU actually generates the Serial RapidIO error/write-port output, which is combined with the error interrupt to generate the error/port-write interrupt.
5. Software reads the queue full bit (IPWSR[QFI]) and determines that the queue full event caused the interrupt. Many events that can generate this interrupt. Software must read several registers to determine that the interrupt is due to a port-write.
6. Software processes the port-write queue entry to which the port-write base address registers (IPWQBAR) are pointing.
7. Software sets the clear queue bit (IPWMR[CQ]) to reenable the hardware to receive another port-write.
8. Software clears the queue full interrupt bit (IPWSR[QFI]) by setting IPWSR[QFI].

16.5.3 Port-Write Controller Interrupts

The error/port-write interrupt is used by the port-write controller. This interrupt is used to notify the processor that some type of error event has occurred in a RapidIO port, message controller, doorbell controller or port-write controller. There are many events that can generate this interrupt. For example, the error management extensions use this interrupt to notify that error events have occurred. In the port-write controller the following event can generate this interrupt.

- *Queue Full*. This interrupt event is enabled (IPWMR[QFIE]) and a port-write is received and written to memory. The event causing this interrupt is indicated by IPWSR[QFI]. The interrupt is held until the queue is not full and the IPWSR[QFI] bit is cleared by writing a value of 1 to it.

- *Transaction Error*. An internal error response is received and this interrupt event is enabled (IPWMR[EIE]).

16.5.4 Discarding Port-Writes

While the queue full bit is set or a port-write is being written to memory but has not completed, all received port-writes are discarded. When a port-write is discarded for one of these reasons, the controller sets the port write discarded bit (IPWSR[PWD]). Note that the port-write busy bit (IPWSR[PWB]) indicates that a port-write is being written to memory but has not completed.

16.5.5 Transaction Errors

When an internal error occurs while the port-write controller is writing data to local memory, the controller takes the following actions:

1. Sets the transaction error bit (IPWSR[TE]) and enters the error state.
2. Generates the Serial RapidIO error/write-port interrupt if IPWMR[EIE] is set.

16.5.6 Software Error Handling

When an error occurs and the Serial RapidIO error/write-port interrupt is generated, software takes the following actions:

1. Determines the cause of the interrupt and processes the error.
2. Polls IPWSR[PWB] to verify that the port-write controller has stopped operation.
3. Disables the port-write controller by clearing IPWMR[PWE].
4. Clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).
5. Disables, reinitializes, and reenables the port-write unit before another maintenance port-write can be received.

When an error occurs and the Serial RapidIO error/write-port interrupt is not enabled, software takes the following actions:

1. Polls the status bits (IPWSR[TE]) to determines that an error has occurred.
2. Polls IPWSR[PWB] to verify that the port-write controller has stopped operation.
3. Clears IPWMR[PWE] to disable the port-write controller.
4. Clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).

16.5.7 Hardware Error Handling

Table 16-42 describes the possible hardware error conditions and what occurs when they are detected. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. When an error is detected, no additional error checking

beyond the current level is performed. Port-writes are processed in a pipeline. The first error detected in the processing pipeline updates the error management extension registers. These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port, as described in **Section 16.2.10**, *Errors and Error Handling*, on page 16-48.

Table 16-42. Inbound Port-Write Hardware Errors

Error	Description
Reserved ftype[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** No response. **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Reserved tt encoding[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Large transport size in small transport mode or small transport size in large transport mode[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[TSE] is set. **Status bit set**: Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]. **Queue Entry Written in local memory:** No **Response status:** No response. **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal destination ID[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITTE] is set. **Status bit set**: Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
An incorrect wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56, or 64 bytes), payload size greater than the size defined by wr_size, or not 64-bit aligned when the size is not 4 bytes.[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.

Table 16-42. Inbound Port-Write Hardware Errors

Error	Description
wr_size value reserved[1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[ITD] set. **Status bit set**: Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]. **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Inbound maintenance port-write received and inbound maintenance port-writes are not supported as indicated by DOCAR[PW][1]	**Error checking level:** 1 **Interrupt generated:** Serial RapidIO error/write-port if LTLEECSR[UT] is set. **Status bit set**: Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]. **Queue Entry Written in local memory:** No **Response status:** Error **Logical/Transport Layer Capture Register:** Updated with packet.[2] **Comments:** Packet is ignored and discarded.
Not an error. An inbound port-write packet with a RapidIO priority of 3	**Error checking level:** 2 **Interrupt generated:** **Status bit set**: **Queue Entry Written in local memory:** Yes **Response status:** No response **Logical/Transport Layer Capture Register:** Inbound port write considered priority 2 by the inbound port write controller since response from memory is required at priority 3. **Comments:**
Port-write controller disabled and port-write received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Port-write controller enabled but in the error state and port-write received	**Error checking level:** 2 **Interrupt generated:** No **Status bit set**: None **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** **Comments:** Packet is ignored and discarded.
Internal error during the write of the port-write queue entry to memory	**Error checking level:** 3 **Interrupt generated:** Serial RapidIO error/write-port if IPWMR[EIE] is set. **Status bit set**: Transaction error in the Port-write status register (IPWSR[TE]). Port-write Failed in the Port-write and Doorbell CSR (PWDCSR[PFA]). **Queue Entry Written in local memory:** No **Response status:** No response **Logical/Transport Layer Capture Register:** **Comments:** Port-write controller stops after the current port-write operation completes.

Table 16-42. Inbound Port-Write Hardware Errors

Error	Description
Notes: **1.** These error types are actually detected in the RapidIO port, not in the port-write controller. **2.** In small transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 78–79). • LTLACCSR[A] gets the address (packet bits 48–76). • LTLDIDCCSR[MDID] gets 0. • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[MSID] gets 0. • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24–31). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 32–35). • LTLCCCSR[MI] gets 0. In large transport size configuration using the packet, the following allocations are made: • LTLACCSR[XA] gets the extended address (packet bits 94–95). • LTLACCSR[A] gets the address (packet bits 64– 92). • LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16–23). • LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24–31). • LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32–39). • LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40–47). • LTLCCCSR[FT] gets the ftype (packet bits 12–15). • LTLCCCSR[TT] gets the ttype (packet bits 48 –51). • LTLCCCSR[MI] gets 0.	

Table 16-43 lists the port-write programming errors.

Table 16-43. Inbound Port-Write Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Port-write queue entry written to non-existent memory	No	No	When a write to memory occurs, the memory controller causes its own interrupt and update its own capture registers. An internal error response is returned. When the port-write controller receives the error response it sets the transaction error bit (IPWSR[TE]) and enters the error state.

16.5.8 Disabling and Enabling the Port-Write Controller

When the port-write controller is disabled by clearing IPWMR[PWE], the following actions are taken:

1. Queue full clears (IPWSR[QF]).
2. Port-write busy (IPWSR[PWB]) clears after a pending port-write queue entry write completes.

Before the port-write controller is reenabled (IPWMR[PWE]), the port-write busy bit (IPWSR[PWB]) must be cleared.

16.5.9 RapidIO Message Passing Logical Specification Registers

The port-write and doorbell command and status register (PWDCSR) includes several port-write controller status bits. These read-only status bits only indicate the state of the port-write controller. See **Section 16.6.9**, *Port Write and Doorbell Command and Status Register (PWDCSR)*, on page 16-141 for details.

- *Available (PWDCSR[PA])*. Indicates that the port-write controller is enabled (IPWnMR[PWE]), the only port-write queue entry is available to be written (IPWnSR[QF]) = 0 and the port-write controller is not in the internal error state (IPWnSR[TE] = 0)
- *Full (PWDCSR[PFU])*. This bit reflects the state of the queue full status bit (IPWnSR[QF])
- *Empty (PWDCSR[PEM])*. This bit always a 1 since a port-write cannot be generated
- *Busy (PWDCSR[PB])*. This bit reflects the state of the busy bit IPWnSR[PWB]
- *Failed (PWDCSR[PFA])*. This bit reflects the state of the transaction error status bit IPWnSR[TE]
- *Error (PWDCSR[PE])*. This bit is always a 0

16.6 RapidIO Programming Model

The RapidIO registers are listed in **Table 16-44**.

Table 16-44. RapidIO Registers

Group	Register	Offset	Acronym	Page
Architectural Registers:	Device Identity Capability Register	0x00000	DIDCAR	**page 16-133**
	Device Information Capability Register	0x00004	DICAR	**page 16-133**
	Assembly Identity Capability Register	0x00008	AIDCAR	**page 16-134**
	Assembly Information Capability Register	0x0000C	AICAR	**page 16-134**
	Processing Element Features Capability Register	0x00010	PEFCAR	**page 16-135**
	Source Operations Capability Register	0x00018	SOCAR	**page 16-136**
	Destination Operations Capability Register	0x0001C	DOCAR	**page 16-138**
	Mailbox Command and Status Register	0x00040	MCSR	**page 16-139**
	Port -Write and Doorbell Command and Status Register	0x00044	PWDCSR	**page 16-141**
	Processing Element Logical Layer Control Command and Status Register	0x0004C	PELLCCSR	**page 16-142**
	Local Configuration Space Base Address 1 Command and Status Register	0x0005C	LCSBA1CSR	**page 16-143**
	Base Device ID Command and Status Register	0x00060	BDIDCSR	**page 16-144**
	Host Base Device ID Lock Command and Status Register	0x00068	HBDIDLCSR	**page 16-145**
	Component Tag Command and Status Register	0x0006C	CTCSR	**page 16-146**

Table 16-44. RapidIO Registers (Continued)

Group	Register	Offset	Acronym	Page
Extended Features Space: 1x/4x LP-Serial	Port Maintenance Block Header 0 (0)	0x00100	PMBH0	**page 16-147**
	Port Link Time-Out Control Command and Status Register ()	0x00120	PLTOCCSR	**page 16-148**
	Port Response Time-out Control Command and Status Register ()	0x00124	PRTOCCSR	**page 16-149**
	Port General Control Command and Status Register	0x0013C	PGCCSR	**page 16-150**
	Port 0–1 Link Maintenance Request Command and Status Register	0x00140 0x00160	PnLMREQCSR	**page 16-151**
	Port 0–1 Link Maintenance Response Command and Status Register	0x00144 0x00164	PnLMRESPCSR	**page 16-152**
	Port 0–1 Local ackID Status Command and Status Register	0x00148 0x00168	PnLASCR	**page 16-153**
	Port 0–1 Error and Status Command and Status Register	0x00158 0x00178	PnESCSR	**page 16-154**
	Port 0–1 Control Command and Status Register	0x0015C 0x0017C	PnCCSR)	**page 16-156**
Extended Features Space: Error Reporting (Logical)	Error Reporting Block Header	0x00600	ERBH	**page 16-158**
	Logical/Transport Layer Error Detect Command and Status Register	0x00608	LTLEDCSR	**page 16-158**
	Logical/Transport Layer Error Enable Command and Status Register	0x0060C	LTLEECSR	**page 16-160**
	Logical/Transport Layer Address Capture Command and Status Register	0x00614	LTLACCSR	**page 16-161**
	Logical/Transport Layer Device ID Capture Command and Status Register	0x00618	LTLDIDCCSR	**page 16-162**
	Logical/Transport Layer Control Capture Command and Status Register	0x0061C	LTLCCCSR	**page 16-163**
Extended Features Space: Error Reporting (Physical)	Port 0–1 Error Detect Command and Status Register	0x00640 0x00680	PnEDCSR	**page 16-164**
	Port 0–1 Error Rate Enable Command and Status Register	0x00644 0x00684	PnERECSR	**page 16-165**
	Port 0–1 Error Capture Attributes Command and Status Register	0x00648 0x00688	PnECACSR	**page 16-166**
	Port 0–1 Packet/Control Symbol Error Capture Command and Status Register	0x0064C 0x0068C	PnPCSECCSR0	**page 16-168**
	Port 0–1 Packet Error Capture Command and Status Register 1	0x00650 0x00690	PnPECCSR1	**page 16-169**
	Port 0–1 Packet Error Capture Command and Status Register 2	0x00654 0x00694	PnPECCSR2	**page 16-170**
	Port 0–1 Packet Error Capture Command and Status Register 3	0x00658 0x00698	PnPECCSR3	**page 16-171**
	Port 0–1 Error Rate Command and Status Register	0x00668 0x006A8	PnERCSR	**page 16-172**
	Port 0–1 Error Rate Threshold Command and Status Register	0x0066C 0x006AC	PnERTCSR	**page 16-173**

Table 16-44. RapidIO Registers (Continued)

Group	Register	Offset	Acronym	Page
Implementation Space: General	Logical Layer Configuration Register	0x10004	LLCR	page 16-174
	Error /Port-write Interrupt Status Register	0x10010	EPWISR	page 16-175
	Logical Retry Error Threshold Configuration Register	0x10020	LRETCR	page 16-176
	Physical Retry Error Threshold Configuration Register ()	0x10080	PRETCR	page 16-177
	Port 0–1 Alternate Device ID Command and Status Register	0x10100 0x10180	PnADIDCSR	page 16-178
	Port 0–1 Pass-Through Accept-All Configuration Register	0x10120 0x101A0	PnPTAACR	page 16-179
	Port 0–1 Logical Outbound Packet Time-to-Live Configuration Register ()	0x10124 0x101A4	PnLOPTTLCR	page 16-180
	Port 0–1 Implementation Error Command and Status Register	0x10130 0x101B0	PnIECSR	page 16-181
	Port 0–1 Serial Link Command and Status Register	0x10158 0x101D8	PnSLCSR	page 16-182
	Port 0–1 Serial Link Error Injection Configuration Register	0x10160 0x101E0	PnSLEICR	page 16-183
Implementation Space: Revision Control	IP Block Revision Register 1	0x10BF8	IPBRR1	page 16-184
	IP Block Revision Register 2	0x10BFC	IPBRR2	page 16-184
Outbound ATMU	Port 0–1 RapidIO Outbound Window Translation Address Register x (offset is x*0x20)	0x10C00 0x10E00	PnROWTARx	page 16-185
	Port 0–1 RapidIO Outbound Window Translation Extended Address Register x (offset is x*0x20)	0x10C04 0x10E04	PnROWTEARx	page 16-186
	Port 0–1 RapidIO Outbound Window Base Address Register x (offset is x*0x20)	0x10C08 0x10E08	PnROWBARx	page 16-187
	Port 0–1 RapidIO Outbound Window Attributes Register x (offset is x*0x20)	0x10C10 0x10E10	PnROWARx	page 16-188
	Port 0–1 RapidIO Outbound Window Segment x (1–3) Registers y (1–8) (offset is (x –1)*0x20 + (y – 1)*0x20)	0x10C34 0x10E34	PnROWSxRy	page 16-190
Inbound ATMU	Port 0–1 RapidIO Inbound Window Translation Address Registers x (offset is (4 – x)*0x20);	0x10D60 0x10F60	PnRIWTARx	page 16-191
	Port 0–1 RapidIO Inbound Window Base Address Registers x (offset is (4 – x)*0x20)	0x10D68 0x10F68	PnRIWBARx	page 16-192
	Port 0–1 RapidIO Inbound Window Attributes Registers x (offset is (4 – x)*0x20)	0x10D70 0x10F70	PnRIWARx	page 16-193

Table 16-44. RapidIO Registers (Continued)

Group	Register	Offset	Acronym	Page
Outbound Message Unit Registers	Outbound Message x Mode Registers (offset is x*0x100)	0x13000	OMxMR	page 16-194
	Outbound Message x Status Registers (offset is x*0x100)	0x13004	OMxSR	page 16-196
	Outbound Message x Descriptor Queue Dequeue Pointer Address Registers (offset is x*0x100)	0x1300C	OMxDQDPAR	page 16-198
	Outbound Message x Source Address Registers (offset is x*0x100)	0x13014	OMxSAR	page 16-199
	Outbound Message x Destination Port Registers (offset is x*0x100)	0x13018	OMxDPR	page 16-200
	Outbound Message x Destination Attributes Registers (offset is x*0x100)	0x1301C	OMxDATR	page 16-201
	Outbound Message x Double-Word Count Registers (offset is x*0x100)	0x13020	OMxDCR	page 16-202
	Outbound Message x Descriptor Queue Enqueue Pointer (offset is x*0x100) Address Registers (offset is x*0x100)	0x13028	OMxDQEPAR	page 16-203
	Outbound Message x Retry Error Threshold (offset is x*0x100) Configuration Registers	0x1302C	OMxRETCR	page 16-204
	Outbound Message x Multicast Group Registers (offset is x*0x100)	0x13030	OMxMGR	page 16-205
	Outbound Message x Multicast List Registers (offset is x*0x100)	0x13034	OMxMLR	page 16-206
Inbound Message Unit Registers	Inbound Message x Mode Registers (offset is x*0x100)	0x13060	IMxMR	page 16-207
	Inbound Message x Status Registers (offset is x*0x100)	0x13064	IMxSR	page 16-209
	Inbound Message x Frame Queue Dequeue Pointer Address Registers (offset is x*0x100)	0x1306C	IMxFQDPAR	page 16-211
	Inbound Message x Frame Queue Enqueue Pointer Address Registers (offset is x*0x100)	0x13074	IMxFQEPAR	page 16-212
	Inbound Message x Maximum Interrupt Report Interval Registers(offset is x*0x100)	0x13078	IMxMIRIR	page 16-213
Outbound Doorbell Unit Registers	Outbound Doorbell Mode Register	0x13400	ODMR	page 16-214
	Outbound Doorbell Status Register	0x13404	ODSR	page 16-215
	Outbound Doorbell Destination Port Register	0x13418	ODDPR	page 16-216
	Outbound Doorbell Destination Attributes Register	0x1341C	ODDATR	page 16-217
	Outbound Doorbell Retry Error Threshold Configuration Register	0x1342C	ODRETCR	page 16-218
Inbound Doorbell Unit Registers	Inbound Doorbell Mode Register	0x13460	IDMR	page 16-219
	Inbound Doorbell Status Register	0x13464	IDSR	page 16-221
	Inbound Doorbell Queue Dequeue Pointer Address Register	0x1346C	IDQDPAR	page 16-222
	Inbound Doorbell Queue Enqueue Pointer Address Register	0x13474	IDQEPAR	page 16-223
	Inbound Doorbell Maximum Interrupt Report Interval Register	0x13478	IDMIRIR	page 16-224
Port-Write Registers	Inbound Port-Write Mode Register	0x134E0	IPWMR	page 16-225
	Inbound Port-Write Status Register	0x134E4	IPWSR	page 16-226
	Inbound Port-Write Queue Base Address Register	0x134EC	IPWQBAR	page 16-227

Note: The base address for the RapidIO registers is: 0xFFF80000.

16.6.1 Device Identity Capability Register (DIDCAR)

DIDCAR Device Identity Capability Register Offset 0x00000

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

Table 16-45. DIDCAR Field Descriptions

Bits	Reset	Description
DI 31–16	0x181A	**Device Identity** Uniquely identifies the type of device from the vendor specified by DVI. The values for DI are assigned and managed by the respective vendor. The value for the MSC8156E = 0x181A.
DVI 15–0	0x0002	**Device Vendor Identity** Identifies the vendor that manufactures 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 value for Freescale Semiconductor, Inc. is 0x0002.

16.6.2 Device Information Capability Register (DICAR)

DICAR Device Information Capability Register Offset 0x00004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DR															
TYPE	R															
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**
	DR															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-46. DICAR Field Descriptions

Bit	Reset	Description
DR 31–0	0x00000000	**Device Revision** Identifies the revision level of the device. The value for DR is assigned and managed by the vendor specified by DIDCAR[DVI].

16.6.3 Assembly Identity Capability Register (AIDCAR)

AIDCAR Assembly Identity Capability Register Offset 0x00008

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	AI															
TYPE	R															
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**
	AVI															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-47. AIDCAR Field Descriptions

Bit	Name	Description
AI 31–16	0x0000	**Assembly Identity** Uniquely identifies the type of assembly from the vendor specified by AVI. The values for AI are assigned and managed by the respective vendor.
AVI 15–0	0x0000	**Assembly Vendor Identity** Identifies the vendor that manufactures 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.

16.6.4 Assembly Information Capability Register (AICAR)

AICAR Assembly Information Capability Register Offset 0x0000C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	AR															
TYPE	R															
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**
	EFP															
TYPE	R															
RESET	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

AICAR contains additional information on the assembly and the pointer to the first entry in the extended features list.

Table 16-48. AICAR Field Descriptions

Bit	Reset	Description
AR 31–16	0x0000	**Assembly Revision**
EFP 15–0	0x0100	**Extended Features Pointer**

16.6.5 Processing Element Features Capability Register (PEFCAR)

PEFCAR Processing Element Features Capability Register Offset 0x00010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	BR	MEM	PROC	SW	—				MB				DB	—		
TYPE	R															
RESET	1	1	1	0	0	0	0	0	1	1	1	1	1	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—											CTLS	EF	EAS		
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

PEFCAR identifies the major functionality provided by the processing element.

Table 16-49. PEFCAR Field Descriptions

Bit	Reset	Description	Settings
BR 31	1	**Bridge** Specifies whether the MSC8156E can bridge to another interface.	
MEM 30	1	**Memory** Specifies whether the MSC8156E has physically addressable local address space and can be accessed as an endpoint through non-maintenance operations.	
PROC 29	1	**Processor** Specifies whether the MSC8156E contains a local processor that executes code.	
SW 28	0	**Switch** The MSC8156E does not bridge to another external RapidIO interface. (SW=0)	
— 27–24	0	Reserved. Write to zero for future compatibility.	
MB 23–20	0b1111	**Mailbox** Specifies the inbound mailboxes supported by the MSC8156E.	1000 Mailbox 0, supported by MSC8156E. 0100 Mailbox 1, supported by MSC8156E. 0010 Mailbox 2, supported by MSC8156E. 0001 Mailbox 3, supported by MSC8156E. 1111 Mailboxes 0–3.
DB 19	1	**Doorbell** Specifies whether the RapidIO Controller supports inbound Doorbells.	
— 18–5	0	Reserved. Write to zero for future compatibility.	
CTLS 4	1	**Common Transport Large Systems** Specifies whether the RapidIO Controller supports large systems. This is configured at power-up through the reset configuration word.	0 Only common transport small systems (up to 256 devices). 1 Large systems up to 65,536 devices.

Table 16-49. PEFCAR Field Descriptions

Bit	Reset	Description	Settings
EF 3	1	**Extended Features** Specifies whether the extended features pointer is valid.	
EAS 2–0	0b001	**Extended Address Support** Indicates the number of address bits supported by the RapidIO controller both as a source and target of an operation.	000 Reserved. 010 Reserved. 001 34-bit addresses.

16.6.6 Source Operations Capability Register (SOCAR)

SOCAR Source Operations Capability Register 0x00018

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	—					
TYPE	R															
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**
	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
TYPE	R															
RESET	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

SOCAR defines the set of RapidIO logical operations that can be issued by the MSC8156E.

Table 16-50. SOCAR Field Descriptions

Bit	Reset	Description
R 31	0	**Read Operation** MSC8156E does not support.
IR 30	0	**IRead Operation** MSC8156E does not support.
RO 29	0	**Read-to-Own Operation** MSC8156E does not support.
DCI 28	0	**Data Cache Invalidate Operation** MSC8156E does not support.
C 27	0	**Castout Operation** MSC8156E does not support.
F 26	0	**Flush Operation** MSC8156E does not support.
IOR 25	0	**I/O-Read Operation** MSC8156E does not support.
ICI 24	0	**Instruction Cache Invalidate Operation** MSC8156E does not support.
TIE 23	0	**TLBIE Operation** MSC8156E does not support.
TIES 22	0	**TLBSYNC Operation** MSC8156E does not support.
— 21–16	0	Reserved. Write to zero for future compatibility.
NR 15	1	**NREAD Operation** MSC8156E supports operation.

Table 16-50. SOCAR Field Descriptions (Continued)

Bit	Reset	Description
NW 14	1	**NWRITE Operation** MSC8156E supports operation.
SW 13	1	**SWRITE Operation** MSC8156E supports operation.
NWR 12	1	**NWRITE_R Operation** MSC8156E supports operation.
M 11	1	**Message Operation**
D 10	1	**Doorbell Operation**
— 9	—	Reserved. Write to zero for future compatibility.
ATS 8	0	**Atomic-Test-and-Swap Operation** MSC8156E does not support.
AI 7	0	**Atomic-Inc Operation** MSC8156E does not support.
AD 6	0	**Atomic-Dec Operation** MSC8156E does not support.
AS 5	0	**Atomic-Set Operation** MSC8156E does not support.
AC 4	0	**Atomic-Clr Operation** MSC8156E does not support.
— 3	0	Reserved. Write to zero for future compatibility.
PW 2	0	**Port-Write Operation** MSC8156E does not support.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.6.7 Destination Operations Capability Register (DOCAR)

DOCAR Destination Operations Capability Register 0x0001C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	—					
TYPE	R															
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**
	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
TYPE	R															
RESET	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0

DOCAR defines the set of RapidIO I/O operations that the MSC8156E can support as a target.

Table 16-51. DOCAR Field Descriptions

Bit	Reset	Description
R 31	0	**Read Operation** MSC8156E does not support.
IR 30	0	**IRead Operation** MSC8156E does not support.
RO 29	0	**Read-to-Own Operation** MSC8156E does not support.
DCI 28	0	**Data Cache Invalidate Operation** MSC8156E does not support.
C 27	0	**Castout Operation** MSC8156E does not support.
F 26	0	**Flush Operation** MSC8156E does not support.
IOR 25	0	**I/O-Read Operation** MSC8156E does not support.
ICI 24	0	**Instruction Cache Invalidate Operation** MSC8156E does not support.
TIE 23	0	**TLBIE Operation** MSC8156E does not support.
TIES 22	0	**TLBSYNC Operation** MSC8156E does not support.
— 21–16	0	Reserved. Write to zero for future compatibility.
NR 15	1	**Nread Operation** Supported.
NW 14	1	**Nwrite Operation** Supported.
SW 13	1	**Swrite Operation** Supported.
NWR 12	1	**Nwrite_R Operation** Supported.
M 11	1	**Message Operation** Supported.
D 10	1	**Doorbell Operation** Supported.

Table 16-51. DOCAR Field Descriptions (Continued)

Bit	Reset	Description
— 9	—	Reserved. Write to zero for future compatibility.
ATS 8	0	**Atomic-Test-and-Swap Operation** MSC8156E does not support.
AI 7	0	**Atomic-Inc Operation** MSC8156E does not support.
AD 6	0	**Atomic-Dec Operation** MSC8156E does not support.
AS 5	0	**Atomic-Set Operation** MSC8156E does not support.
AC 4	0	**Atomic-Clr Operation** MSC8156E does not support.
— 3	0	Reserved. Write to zero for future compatibility.
PW 2	1	**Port-Write Operation** Supported.
— 1–0	0	Reserved. Write to zero for future compatibility.

16.6.8 Mailbox Command and Status Register (MCSR)

MCSR Mailbox Command and Status Register 0x00040

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	A0	FU0	EM0	B0	FA0	ERR0	—		A1	FU1	EM1	B1	FA1	ERR1	—	
TYPE	R															
RESET	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MCSR reflects the status of the RapidIO message controllers.

Table 16-52. MCSR Field Definitions

Bits	Reset	Description	Settings
A0 31	0	**Available** Specifies whether message controller 0 is initialized and ready to accept messages. When this bit is cleared, all incoming message transactions to message controller 0 return error responses.	0 Not ready. 1 Ready.
FU0 30	0	**Full** Specifies whether message controller 0 is full. When FU0 is set, new messages to message controller 0 are retried.	0 Not full. 1 Full.
EM0 29	1	**Empty** Specifies whether message controller 0 contains outstanding messages.	0 Contains outstanding messages. 1 Contains no outstanding messages.

Table 16-52. MCSR Field Definitions (Continued)

Bits	Reset	Description	Settings
B0 28	0	**Busy** Specifies whether message controller 0 is busy processing a message. When this bit is set, new message operations to message controller 0 return retry responses.	0 Not busy. 1 Busy.
FA0 27	0	**Failure** Specifies whether message controller 0 has encountered an internal error and is awaiting assistance. When this bit is set, all incoming message transactions to message controller 0 return error responses.	0 No internal error. 1 Internal fault or error condition encountered.
ERR0 26	0	**Error** This field always returns a value of 0.	
— 25–24	0	Reserved. Write to zero for future compatibility.	
A1 23	0	**Available** Specifies whether message controller 1 is initialized and ready to accept messages. When this bit is cleared, all incoming message transactions to message controller 1 return error responses.	0 Not ready. 1 Ready.
FU1 22	0	**Full** Specifies whether message controller 1 is full. When FU0 is set, new messages to message controller 1 are retried.	0 Not full. 1 Full.
EM1 21	1	**Empty** Specifies whether message controller 1 contains outstanding messages.	0 Contains outstanding messages. 1 Contains no outstanding messages.
B1 20	0	**Busy** Specifies whether message controller 1 is busy processing a message. When this bit is set, new message operations to message controller 1 return retry responses.	0 Not busy. 1 Busy.
FA1 19	0	**Failure** Specifies whether message controller 1 has encountered an internal error and is awaiting assistance. When this bit is set, all incoming message transactions to message controller 1 return error responses.	0 No internal error. 1 Internal fault or error condition encountered.
ERR1 18	0	**Error** This field always returns a value of 0.	
— 17–0	0	Reserved. Write to zero for future compatibility.	

16.6.9 Port Write and Doorbell Command and Status Register (PWDCSR)

PWDCSR Port-Write and Doorbell Command and Status Register Offset 0x00044

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	A	FU	EM	B	FA	ERR	—									
TYPE	R															
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**
	—								PA	PFU	PEM	PB	PFA	PE	—	
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

PWDCSR reflects the status of the RapidIO doorbell and port-write hardware. For details, refer to the *RapidIO Interconnect Specification, Revision 1.2* in sections “Doorbell CSR” and “Write Port CSR.”:

Table 16-53. PWDCSR Field Descriptions

Bits	Reset	Description	Settings
A 31	0	**Available** Specifies whether the doorbell unit is available and ready to accept doorbell messages. When this bit is cleared, all incoming doorbell transactions return error responses.	0 Not ready. 1 Ready.
FU 30	0	**Full** Specifies whether the doorbell unit is full. When this bit is set, new doorbell messages are retried.	0 Not full. 1 Full.
EM 29	1	**Empty** Specifies whether the doorbell unit has outstanding doorbell messages.	0 Outstanding doorbell messages. 1 No outstanding doorbell messages.
B 28	0	**Busy** Specifies whether the doorbell unit is busy processing a doorbell message. When this bit is set, incoming transactions are not retried.	0 Not busy. 1 Busy.
FA 27	0	**Failure** Specifies whether the doorbell unit has encountered an internal fault or error condition and is awaiting assistance. When this bit is set, all incoming doorbell transactions return error responses.	0 No internal error. 1 Internal fault or error condition encountered.
ERR 26	0	**Error** This field always returns a value of 0.	
— 25–8	0	Reserved. Write to zero for future compatibility.	
PA 7	0	**Port-Write Unit Available** Specifies whether the port-write unit is available and ready to accept a port-write packet. When this bit is cleared, all incoming port-write packets are discarded.	0 Not available. 1 Ready.
PFU 6	0	**Port-Write Unit Full** Specifies whether the port-write unit is full. When this bit is set, all incoming port-write packets are discarded.	0 Not full. 1 Full.
PEM 5	1	**Port-Write Unit Empty** This field always returns a value of 1.	

Table 16-53. PWDCSR Field Descriptions (Continued)

Bits	Reset	Description	Settings
PB 4	0	**Port-write Unit Busy** Specifies whether the port-write unit is busy processing a port-write packet. When this bit is set, incoming port-write packets are discarded.	0 Not busy. 1 Busy.
PFA 3	0	**Failure** Specifies whether the port-write unit has encountered an internal fault or error condition and is awaiting assistance. When this bit is set, all incoming port-write transactions are discarded.	0 No internal error. 1 Internal fault or error condition encountered.
PE 22	0	**Error** This field always returns a value of 0.	
— 1–0	0	Reserved. Write to zero for future compatibility.	

16.6.10 Processing Element Logical Layer Control Command and Status Register (PELLCCSR)

PELLCCSR Processing Element Logical Layer Control Command and Status Register — Offset 0x0004C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—														EAC	
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PELLCCSR controls the extended addressing abilities. RapidIO endpoint supports only 34-bit addressing.

Table 16-54. PELLCCSR Field Descriptions

Bit	Reset	Description
— 31–3	0	Reserved. Write to zero for future compatibility.
EAC 2–0	0b001	**Extended Addressing Control** The read-only value of 0b001 specifies 34-bit addresses.

16.6.11 Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

LCSBA1CSR Local Configuration Space Base Address 1 Command and Status Register — Offset 0x0005C

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

LCSBA1CSR specifies the most significant bits of the local physical address double-word offset for the MSC8156E configuration register space allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of an MSC8156E 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 with all registers, an external processor writing to LCSBA1CSR should not assume that the write occurs until a response is received.

Table 16-55. LCSBA1CSR Field Descriptions

Bit	Reset	Description
— 31	0	Reserved. Write to zero for future compatibility.
LCSBA 30–17	0x1FFE	Local configuration space base address. These bits correspond to the highest 14 bits of the 34-bit RapidIO address space.
— 16–0	0	Reserved. Write to zero for future compatibility.

16.6.12 Base Device ID Command and Status Register (BDIDCSR)

BDIDCSR Base Device ID Command and Status Register Offset 0x00060

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

BDIDCSR contains the base device ID values for the processing element.

Table 16-56. BDIDCSR Field Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
BDID 23–16	0xFF	**Base Device ID in Small Common Transport System (RapidIO Device ID)** Valid only if PEFCAR[CTLS] is cleared. If the RapidIO controller is configured as a host, the highest 5 bits of BDID are cleared and the lowest 3 bits are equal to the lowest three bits of the RCWHR[DEVID] field (see **Section 5.3.2**, *Reset Configuration Word High Register (RCWHR)*, on page 5-19). If the RapidIO controller is configured as an agent, BDID = 0xFF.
LBDID 15–0	0xFFFF	**Large Base Device ID in Large Common Transport System** Valid only if PEFCAR[CTLS] is set. If the RapidIO controller is configured as a host, the highest 13 bits of LBDID are cleared and the lowest 3 bits are equal to the lowest three bits of the RCWHR[DEVID] field (see **Section 5.3.2**, *Reset Configuration Word High Register (RCWHR)*, on page 5-19). If the RapidIO controller is configured as an agent, LBDID = 0xFFFF.

16.6.13 Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

HBDIDLCSR Host Base Device ID Lock Command and Status Register — Offset 0x00068

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	HBDID															
TYPE	R															
RESET	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HBDIDLCSR contains the base device ID value for the processing element in the system that initializes this device. The HBDID field is a write-once/resettable field with a lock function. When HBDID is written, all subsequent writes to the field are ignored, except when the value written matches the value in the field. Then, the register is reinitialized to 0xFFFF. After HBDID is written, the processing element must read the host base device ID lock CSR to verify that it owns the lock before it attempts to initialize the device.

Table 16-57. HBDIDLCSR Field Descriptions

Bit	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
HBDID 15–0	0xFFFF	**Host Base Device ID** The host base device ID for the processing element that initializes this device. Only the first write to this field is accepted; all other writes are ignored, except when the value written matches the value contained in the field. In this case, the register is reinitialized to 0xFFFF.

16.6.14 Component Tag Command and Status Register (CTCSR)

CTCSR Component Tag Command and Status Register Offset 0x0006C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CT															
TYPE	R/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**
	CT															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CTCSR contains a component tag value for the processing element that software can assign when the device is initialized. It is unused internally in RapidIO Endpoint.

Table 16-58. CTCSR Field Descriptions

Bit	Reset	Description
CT 31–0	0	**Component Tag**

16.6.15 Port Maintenance Block Header 0 (PMBH0)

PMBH0 Port Maintenance Block Header 0 Offset 0x00100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EFPTR															
TYPE	R															
RESET	0	0	0	0	0	1	1	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**
	EFID															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PMBH0 contains a pointer (EFPTR) to the next extended features space, error management block, extended features block (EFBLK), and the EFID that identifies it as the generic endpoint port maintenance block header. While registers defined by software-assisted error recovery are supported, software-assisted error recovery is not. Therefore, the RapidIO Endpoint is defined here as not supporting software-assisted error recovery.

Table 16-59. PMBH0 Field Descriptions

Bit	Reset	Description
EFPTR 31–15	0x0600	**Extended Features Pointer**
EFID 16–31	0x0001	**Extended Features ID**

16.6.16 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

PLTOCCSR Port Link Time-Out Control Command and Status Register — Offset 0x00120

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

PLTOCCSR contains the link time-out value for all ports. This time-out is for link events such as sending a packet to receive the corresponding acknowledge and sending a link-request to receive the corresponding link-response. The reset value is the maximum time-out interval and represents between 3 and 5 seconds.

The time-out increment unit is a function of the OCN bus clock frequency (the HSSI clock frequency) and the value given in the RapidIO prescaler field (CLK_GPR0[RPTE]) and is within the range of 255 ns (±5 ns). The exact formula to calculate this value is:

$$\text{Timer\_Resolution} = 2 \times (\text{RPTE} + 1) / \text{ocn\_clk\_freq}$$

where ocn_clk_freq = HSSI clock frequency

RPTE = Decimal value of CLK_GPR0[RPTE]

For example, if the OCN clock frequency = 333 MHz (RCWLR[MODCK] = 0) and CLK_GPR0[RPTE] = 41, the timer resolution is 252 ns.

Table 16-60. PLTOCCSR Field Descriptions

Bit	Reset	Description
TV 31–8	0xFFFFFF	**Time-Out Value** Clearing this field to all zeros disables the link time-out timer. This value is loaded each time the link time-out timer starts.
— 7–0	0x01	Reserved. Write to 0x01 for future compatibility.

16.6.17 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

PRTOCCSR Port Response Time-Out Control Command and Status Register — Offset 0x00124

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

PRTOCCSR contains the time-out timer count for all ports. This time-out is for sending a request packet to receive the corresponding response packet.The reset value is the maximum time-out interval and represents between 3 and 5 seconds. This applies to RapidIO Endpoint and the messaging unit.

The time-out increment unit is a function of the OCN bus clock frequency (the HSSI clock frequency) and the value given in the RapidIO prescaler field (CLK_GPR0[RPTE]) and is within the range of 255 ns (±5 ns). The exact formula to calculate this value is:

$$\text{Timer\_Resolution} = 2 \times (\text{RPTE} + 1) / \text{ocn\_clk\_freq}$$

where ocn_clk_freq = HSSI clock frequency

RPTE = Decimal value of CLK_GPR0[RPTE]

For example, if the OCN clock frequency = 333 MHz (RCWLR[MODCK] = 0) and CLK_GPR0[RPTE] = 41, the timer resolution is 252 ns.

Table 16-61. PRTOCCSR Field Descriptions

Bit	Reset	Description
TV 31–8	0xFFFFFF	**Time-Out Value** Clearing this field to all zeros disables the response time-out timer. This value is loaded each time the response time-out timer starts.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.18 Port General Control Command and Status Register (PGCCSR)

PGCCSR Port General Control Command and Status Register Offset 0x0013C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	H	M	D	—												
TYPE	R/W															
RESET	x	x	x	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**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PGCCSR contains control register bits for the RapidIO interface.

Note: The user must initialize the value of M to 1. Otherwise, no outbound transactions are initiated by the MSC8156E, including messages and doorbells.

Table 16-62. PGCCSR Field Descriptions

Bit	Reset	Description	Settings
H 31	RCWHR [RHE]	**Host** Determines the host/agent configuration for the device. Notice that by default H = 0.	0 Agent device. 1 Host device.
M 30	RCWHR [RHE]	**Master** Clearing this bit (M = 0) disables all outbound transactions and prevents sending any outbound packets.	0 Device is not enabled to send requests to the system. 1 Device is enabled to send requests to the system.
D 29	RDWHR [RHE]	**Discovered** Indicates whether the device is discovered by the system host.	0 Device not discovered by system host. 1 Device is discovered by system host.
— 28–0	0	Reserved. Write to zero for future compatibility.	

16.6.19 Port 0–1 Link Maintenance Request Command and Status Register (PnLMREQCSR)

P0LMREQCSR Port 0–1 Link Maintenance Request Offset 0x00140
P1LMREQCSR Command and Status Register Offset 0x00160

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—													C		
TYPE	R													R/W		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Port 0–1 link maintenance request command and status register (PnLMREQCSR), is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the corresponding RapidIO endpoint port interface. Make sure when writing this register that is not used for software-assisted error recovery (which is not supported). **Table 16-63** lists PnLMREQCSR fields.

Table 16-63. PnLMREQCSR Field Descriptions

Bit	Reset	Description
— 31–3	0	Reserved. Write as zero for future compatibility.
C 2–0	0	**Link Request Command** Contains the link request command to send. If read, this field returns the last written value. If written with a value other than 0x011 (reset device) or 0b100 (input status), the resulting operation is undefined. All other values are reserved in the RapidIO specification.

16.6.20 Port 0–1 Link Maintenance Response Command and Status Register (PnLMRESPCSR)

P0LMRESPCSR Port 0–1 Link Maintenance Response Command and Status Register — Offset 0x00144
P0LMRESPCSR Port 0–1 Link Maintenance Response Command and Status Register — Offset 0x00164

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RV	—														
TYPE	R															
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**
	—						AS					LS				
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Port 0–1 link maintenance response command and status register (PnLMRESPCSR), 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. **Table 16-64** lists PnLMRESPCSR fields.

Table 16-64. PnLMRESPCSR Field Descriptions

Bit	Reset	Description	Settings
RV 31	0	**Response Valid** This bit indicates one of two conditions: • If the link-request causes a link-response, a set bit indicates that the link-response was received and the status fields are valid. • If the link-request did not cause a link-response, a set bit indicates that the link-request was transmitted. **Note:** This bit clears on read.	0 Link response not received, link response not valid, or link request was not transmitted. 1 Link response received with valid status bits or link request was transmitted.
— 30–10	0	Reserved. Write as zero for future compatibility.	
AS 9–5	0	**AckID_Status** This field holds the AckID status field from the link response.	
LS 4–0	0	**Link Status** This field holds the link status field from the link response.	

16.6.21 Port 0–1 Local ackID Command and Status Register (PnLASCR)

P0LASCSR Port 0–1 Local ackID Command and Status Register Offset 0x00148
P1LASCSR Offset 0x00168

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—			IA					—							
TYPE	R			R/W					R							
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**
	—			OUTA					—			OBA				
TYPE	R											R/W				
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Port 0–1 local ackID status command and status register (PnLASCSR) is accessible both by the core 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 not to use this register for software error management. **Table 16-65** lists PnLASCSR fields.

Table 16-65. PnLASCSR Field Descriptions

Bit	Reset	Description
— 31–29	0	Reserved. Write as zero for future compatibility.
IA 28–24	0	**Input ackID** The value of the next expected Input port ackID.
23–13	0	Reserved. Write as zero for future compatibility.
OUTA 12–8	0	**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.
— 7–5	0	Reserved. Write as zero for future compatibility.
OBA 4–0	0	**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, a newly-written value will be ignored

16.6.22 Port 0–1 Error and Status Command and Status Register (PnESCSR)

P0ESCSR Port 0 Error Command and Status Register Offset 0x00158
P1ESCSR Offset 0x00178

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—					OPD	OFE	ODE	—			ORE	OR	ORS	OEE	OES
TYPE	R					W1C	W1C	W1C				W1C	R		W1C	R
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**
	—					IRS	IEE	IES	—			PWP	—	PE	PO	PU
TYPE	R						W1C	R						W1C	R	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PnESCSR is accessed when a local master or an external device needs to examine the port error and status information.

Table 16-66. PnESCSR Field Descriptions

Bit	Reset	Description
— 31–27	0	Reserved. Write to zero for future compatibility.
OPD 26	0	**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. • The link-partner does not accept it while the PnERTCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded, PnCCSR[DPE] is set, and PnCCSR[SPF] is not set (and link-response returns the expected ID acknowledge). When OPD is set, it remains set until it is written with a logic 1 to clear it.
OFE 25	0	**Output Fail Encountered** The output port has encountered a failed condition because the error rate counter (PnEERCSR[ERC]) has met or exceeded the port failed error threshold (PnERTCSR[ERFTT]). OFE remains set until it is written with a logic 1 to clear it. When it is cleared, it does not assert again unless the error rate counter dips below the port failed error threshold and then meets or exceeds it again.
ODE 24	0	**Output Degraded Condition Encountered** The output port has encountered a degraded condition because the error rate counter (PnEERCSR[ERC]) has met or exceeded the port degraded error threshold (PnERTCSR[ERDTT]). ODE remains set until it is written with a logic 1 to clear it. When it is cleared, it does not assert again unless the error rate counter dips below the port degraded error threshold and then meets or exceeds it again.
— 23–21	0	Reserved. Write to zero for future compatibility.
ORE 20	0	**Output Retry Condition Encountered** The output port has encountered a retry condition. This bit is set when ORS is set. ORE remains set until it is written with a logic 1 to clear it.
OR 19	0	**Output Retry** The output port has received a packet retry control symbol and cannot make forward progress. OR is set when ORS is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. Read only.

Table 16-66. PnESCSR Field Descriptions (Continued)

Bit	Reset	Description
ORS 18	0	**Output Stop** The output port stops due to a retry. Read only.
OEE 17	0	**Output Error Encounter** The output port encountered (and possibly recovered from) a transmission error. OEE is set when OES is set. It remains set until it is written with a logic 1 to clear it.
OES 16	0	**Output Error Stop** The output port is stopped due to a transmission error. Read only.
— 15–11	0	Reserved. Write to zero for future compatibility.
IRS 10	0	**Input Retry Stop** The input port is stopped due to a retry. Read only.
IEE 9	0	**Input Error Encounter** The input port encountered (and possibly recovered from) a transmission error. IEE is set when IES is set. It remains set until it is written with a logic 1 to clear it.
IES 8	0	**Input Error Stop** The input port is stopped due to a transmission error. Read only.
— 7–5	0	Reserved. Write to zero for future compatibility.
PWP 4	0	**Port Write** Port has encountered a condition requiring it to initiate a maintenance port-write operation. PWP is valid only if the device can issue a maintenance port-write transaction. RapidIO Endpoint cannot issue port-writes. This bit is hard-wired to 0. Read only.
— 3	0	Reserved
PE 2	0	**Port Error** The input or output port has encountered an error from which hardware could not recover. PE remains set until it is written with a logic 1 to clear it. PE indicates that OFE is set while CCSR[SPF] is set. That is, reaching the failed threshold has caused the output port to stop transmitting packets.
PO 1	0	**Ports Operating** The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. Read only.
PU 0	1	**Ports Uninitialized** Input and output ports are not initialized. This bit and PO are mutually exclusive. Read only.

16.6.23 Port 0–1 Control Command and Status Register (PnCCSR)

P0CCSR Port 0–1 Control Command and Status Register Offset 0x0015C
P1CCSR Offset 0x0017C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PW		IPW			PWO			PD	OPE	IPE	ECD	MEP	—		
TYPE	R					R/W							R			
RESET	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—												SPF	DPE	PL	PT
TYPE	R												R/W			R
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PnCCSR contains control register bits for the RapidIO port. This register is for serial implementation only.

Table 16-67. PnCCSR Field Descriptions

Bit	Reset	Description	Settings
PW 31–30	0b01	**Port Width** Specifies the hardware width of the port. Read only.	00 Single-lane port. 01 Four-lane port (the default). 10–11 Reserved.
IPW 29–27	0b010	**Initialized Port Width** Specifies the width of the ports after they are initialized. Read only. If the port degrades to 1x, the value changes to 0b000. In single lane mode, the RapidIO block can synchronize on lane 0 or lane 2. Programming the PWO field to 010 forces the controller to synchronize on lane 0.	000 Single-lane port, lane 0 or 2. 001 Reserved. 010 Four-lane port (the default). 011–111 Reserved.
PWO 26–24	0b000	**Port Width Override** Soft port configuration to override the hardware size. Change PWO only when the port is uninitialized. First disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port.	000 No override (the default). 001 Reserved. 010 Force single lane, lane 0. 011–111 Reserved.
PD 23	0	**Port Disable** When this bit is set, the port does not accept or transmit any transaction. The output will congest if packets are sent to a disabled port.	0 Port receiver/drivers are enabled. 1 Port receiver/drivers are disabled and are unable to receive or transmit.
OPE 22	1	**Output Port Transmit Enable** Specifies whether the port is enabled to issue packets. When OPE is cleared, the port routes or responds to I/O logical maintenance packets. Control symbols are not affected and are sent normally. The initial value of OPE is read from configuration pins.	0 Port is stopped and not enabled to issue packets. 1 Port is enabled to issue packets.

Table 16-67. PnCCSR Field Descriptions (Continued)

Bit	Reset	Description	Settings
IPE 21	1	**Input port Receive Enable** Specifies whether the port is enable to respond to packets. When IPE is cleared, the port can only 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. The value of IPE must equal the value of OPE for the RapidIO controller to function properly. The initial value of IPE is read from configuration pins.	0 Port is stopped and not enabled to respond to packets. 1 Port is enabled to respond to packets.
ECD 20	0	**Error Checking Disable** Disables all RapidIO transmission error checking. This bit is hard-wired to 0.	0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.
MEP 19	0	**Multicast Event Participant** This bit is hard-wired to 0. The MSC8156E does not participate in multicast events.	
— 18–4	0	Reserved. Write to zero for future compatibility.	
SPF 3	0	**Stop on Port Failed Encounter Enable** Used with the DPE bit to force certain behavior when the error rate failed threshold is met or exceeded.	0 Stop on port failed disabled. 1 Stop on port failed enabled.
DPE 2	0	**Drop Packet Enable** Used with the SPF bit to force certain behavior when the error rate failed threshold is met or exceeded.	
PL 1	0	**Port Lockout** Stops the port so that is cannot issue or receive 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 the sending device to signal an error condition.	0 Packets that can be received and issued are controlled by the state of the OPE and IPE bits. 1 The port is stopped and not enabled to issue or receive packets.
PT 0	1	**Port Type** Hard-wired to 1.	0 Reserved. 1 Serial port.

16.6.24 Error Reporting Block Header (ERBH)

ERBH Error Reporting Block Header Offset 0x00600

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EFPTR															
TYPE	R															
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**
	EFID															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

ERBH contains the EFPTR to the next EFBLK. The next EFPTR is 0x0000 since this is the last set of registers in the extended features space. ERBH also contains EFID that identifies this as the error reporting block header.

Table 16-68. ERBH Field Descriptions

Bit	Reset	Description
EFPTR 31–16	0	**Extended Features Pointer** Points to the next EFBLK.
EFID 15–0	0x0007	**Extended Features ID** Identifies this as the error reporting block header.

16.6.25 Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)

LTLEDCSR Logical/Transport Layer Error Detect Command and Status Register Offset 0x00608

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IER	MER	—	MFE	ITD	ITTE	MRT	PRT	UR	UT	—					
TYPE	R/W		R	R/W							R					
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**
	—								IACB	OACB	—	RETE	TSE	PTTL	—	
TYPE	R								R/W		R	R/W			R	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLEDCSR indicates the error that was detected by the logical or transport logic layer. Software writes 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 is 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. An error that is not enabled sets the detect bit in this register as long as a capture has not yet occurred. You can program fields in this

register to emulate a hardware error during software development. Undefined results occur if fields in this register are set while an actual Logical/Transport Layer error is detected. Note that this register can be written with an invalid combination of bits set, and care should be taken to avoid this.

Table 16-69. LTLEDCSR Field Descriptions

Bit	Name	Description
IER 31	0	**I/O Error Response** Indicates an error response for an I/O logical layer request.
MER 30	0	**Message Error Response** Indicates an error response for an MSG logical layer request. The error is detected and captured in the message unit.
— 29	0	Reserved. Write to zero for future compatibility.
MFE 28	0	**Message Format Error** Indicates a MESSAGE packet data payload with an invalid size or segment. The error is detected and captured in the message unit.
ITD 27	0	**Illegal Transaction Decode** Indicates received illegal fields in the request/response packet for a supported transaction (IO/MSG logical)
ITTE 26	0	**Illegal Transaction Target Error** Indicates a packet containing a destination ID that is not defined for this end point when accept-all is not enabled.
MRT 25	0	**Message Request Time-Out** Indicates that a required message request has not been received within the specified time-out interval. The error is detected and captured in message unit.
PRT 24	0	**Packet Response Time-Out** Indicates that a required response was not received within the specified time-out interval (IO/MSG logical)
UR 23	0	**Unsolicited Response** Indicates that an unsolicited/unexpected response packet was received (IO/MSG logical).
UT 22	0	**Unsupported Transaction** Indicates a received transaction that is not supported in the destination operations CAR.
— 21–8	0	Reserved. Write to zero for future compatibility.
IACB 7	0	**Inbound ATMU Crossed Boundary** Indicates a received transaction that crosses an Inbound ATMU boundary.
OACB 6	0	**Outbound ATMU Crossed Boundary** Indicates a received transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsegmented boundary.
— 5	0	Reserved. Write to zero for future compatibility.
RETE 4	0	**Retry Error Threshold Exceeded** Indicates that the allowed number of logical retries (given by LRETCR[RET]) has been exceeded. The message unit also drives RETE when the allowed number of message retries is exceeded.
TSE 3	0	**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 unsupported by this device).
PTTL 2	0	**Packet Time-to-Live Error** Indicates that a packet time-to-live error occurred (that is, a packet could not be successfully transmitted before the packet time-to-live counter expired).
— 1–0	0	Reserved. Write to zero for future compatibility.

16.6.26 Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)

LTLEECSR Logical/Transport Layer Error Enable Command and Status Register — Offset 0x0060C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IER	MER	—	MFE	ITD	ITTE	MRT	PRT	UR	UT	—					
TYPE	R/W		R	R/W							R					
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**
	—								IACB	OACB	—	RETE	TSE	PTTL	—	
TYPE	R								R/W		R	R/W		R/W	R	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLEECSR 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. LTLEECSR is stored in all ports and the message unit.

Table 16-70. LTLEECSR Field Descriptions

Bit	Name	Description
IER 31	0	**I/O Error Response Enable** Enables reporting of an I/O error response. It captures and locks the error.
MER 30	0	**Message Error Response Enable** Enables reporting of a message error response. It captures and locks the error. Capture done in message unit.
— 29	0	Reserved. Write to zero for future compatibility.
MFE 28	0	**Message Format Error Enable** Enables reporting of a message format error. It captures and locks the error. Capture done in messaging unit.
ITD 27	0	**Illegal Transaction Decode Error Enable** Enables reporting of an illegal transaction decode error. It captures and locks the error.
ITTE 26	0	**Illegal Transaction Target Error Enable** Enables reporting of an illegal transaction target error. It captures and locks the error.
MRT 25	0	**Message Request Time-Out Error Enable** Enables reporting of a message request time-out error. It captures and locks the error. Capture done in messaging unit.
PRT 24	0	**Packet Response Time-Out Error Enable** Enables reporting of a packet response time-out error. It captures and locks the error.
UR 23	0	**Unsolicited Response Error Enable** Enables reporting of an unsolicited response error. It captures and locks the error.
UT 22	0	**Unsupported Transaction Error Enable** Enables reporting of an unsupported transaction error. It captures and locks the error.
— 21–8	0	Reserved. Write to zero for future compatibility.
IACB 7	0	**Inbound ATMU Crossed Boundary Error Enable** Enables reporting of a received transaction that crosses an inbound ATMU boundary. It captures and locks the error.
OACB 6	0	**Outbound ATMU Crossed Boundary** Indicates a received transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsegmented boundary.

Table 16-70. LTLEECSR Field Descriptions (Continued)

Bit	Name	Description
— 5	0	Reserved. Write to zero for future compatibility.
RETE 4	0	**Retry Error Threshold Exceeded Enable** Enables reporting when the allowed number of logical retries is exceeded.
TSE 3	0	**Transport Size Error Enable** Enables error reporting when the field is not consistent with the CTLS bit of the processing element features CAR (that is, the tt value is reserved or indicates a common transport system unsupported by this device).
PTTL 2	0	**Packet Time-to-Live Error Enable** Enables reporting of a packet time-to-live error. Capture and lock the result.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.27 Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)

LTLACCSR Logical/Transport Layer Address Capture Command and Status Register — Offset 0x00614

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	A															
TYPE	R/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**
	A													—	XA	
TYPE	R/W													R	R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLACCSR contains error information. It is locked when a logical/transport error is detected and the corresponding enable bit is set. LTLACCSR is stored in the port and in the message unit, although the values in this register can differ between the port and the message unit. The message unit LTLACCSR cannot lock if the port is locked; the port LTLACCSR cannot lock if the message unit is locked.

Note: Fields in this register can be set by writing to the register. You can use this to emulate a hardware error during software development. Undefined results occur if fields in the register are changed while an actual Logical/Transport error is being detected.

Table 16-71. LTLACCSR Field Descriptions

Bit	Reset	Description
A 31–3	0	**Address** Normally, the least significant 29 bits of the address associated with the error (for requests, for responses, if available). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors*, on page 16-53.

Table 16-71. LTLACCSR Field Descriptions

Bit	Reset	Description
— 2	0	Reserved. Write to zero for future compatibility.
XA 1–0	0	**Extended Address MSBs** Normally the extended address bits of the address associated with the error (for requests, responses, if available). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.

16.6.28 Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)

LTLDIDCCSR Logical/Transport Layer Device ID Capture Command and Status Register Offset 0x00618

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DIDMSB								DID							
TYPE	R/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**
	SIDMSB								SID							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LTLDIDCCSR contains error information. It is locked when a logical/transport error is detected and the corresponding enable bit is set. LTLDIDCCSR is stored in the port and in the message unit, although the values in this register can differ between the port and the message unit. The message unit LTLDIDCCSR cannot lock if the port is locked; the port LTLDIDCCSR cannot lock if the message unit is locked.

Note: Fields in this register can be set by writing to the register. You can use this to emulate a hardware error during software development. Undefined results occur if fields in the register are changed while an actual Logical/Transport error is being detected.

Table 16-72. LTLDIDCCSR Field Descriptions

Bit	Reset	Description
DIDMSB 31–24	0	**Destination ID MSB** Normally, the most significant byte of the destination ID associated with the error. This field is valid only if the CTLS bit of the Processing Element Features CAR is set (large transport systems only). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.
DID 23–16	0	**Destination ID** Normally, the destination ID (or least significant byte of the destination ID if large transport system) associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.

Table 16-72. LTLDIDCCSR Field Descriptions

Bit	Reset	Description
SIDMSB 15–8	0	**Source ID MSB** Normally, the most significant byte of the source ID associated with the error. This field is valid only if the CTLS bit of the Processing Element Features CAR is set (large transport systems only). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.
SID 7–0	0	**Source ID** Normally, the source ID (or least significant byte of the source ID if large transport system) associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.

16.6.29 Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)

LTLCCCSR Logical/Transport Layer Control Capture Command and Status Register — Offset 0x0061C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FT				TT				MI							
TYPE	R/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**
	—															
TYPE	R/W															
RESET																

LTLCCCSR contains error information. LTLCCCSR is stored in the port and in the message unit, although the values in this register can differ between the port and the message unit. The message unit LTLCCCSR cannot lock if the port is locked; the port LTLCCCSR cannot lock if the message unit is locked.

Note: Fields in this register can be set by writing to the register. You can use this to emulate a hardware error during software development. Undefined results occur if fields in the register are changed while an actual Logical/Transport error is being detected.

Table 16-73. LTLCCCSR Field Descriptions

Bit	Reset	Description
FT 31–28	0	**Format Type** Normally, the format type associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.
TT 27–24	0	**Transaction Type** Normally, the transaction type associated with the error. For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.
MI 23–16	0	**Message Information** Normally, the message information: letter, mbox, and msgseg for the last message request received for the mailbox with an error (message errors only). For details, see **Section 16.2.10.3**, *Logical Layer RapidIO Errors,* on page 16-53.
— 15–0	0	Reserved. Write to zero for future compatibility.

16.6.30 Port 0–1 Error Detect Command and Status Register (PnEDCSR)

P0EDCSR Port 0–1 Error Detect Command and Status Register Offset 0x00640
P1EDCSR Offset 0x00680

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—									CCS	AUA	PNA	UA	CRC	EM	—
TYPE	R/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**
	—										NOA	PE	—	DE	UCS	LTO
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnEDCSR indicates transmission errors detected by the hardware. Software can write bits in this register with a value of 1 to increment the error rate counter. Undefined results occur if this register is written while actual physical layer errors are detected by the port

Table 16-74. PnEDCSR Field Descriptions

Bit	Reset	Description
— 31–23	0	Reserved. Write to zero for future compatibility.
CCS 22	0	**CRC Control Symbol** Received a control symbol with a bad CRC value.
AUA 21	0	**Acknowledge Control With Unexpected Acknowledge ID** Received an acknowledge control symbol with an unexpected acknowledge ID (packet-accepted or packet-retry).
PNA 20	0	**Packet Not Accepted** Received packet-not-accepted acknowledge control symbol.
UA 19	0	**Unexpected Acknowledge ID** Received packet with unexpected ackID value.
CRC 18	0	**Bad CRC** Received a packet with a bad CRC value.
EM 17	0	**Exceed Maximum** Received packet which exceed the maximum allowed size (276 bytes).
— 16–6	0	Reserved. Write to zero for future compatibility.
NOA 5	0	**Not Outstanding Acknowledge** Link-response received with an ackID that is not outstanding.
PE 4	0	**Protocol Error** An unexpected packet or control symbol was received.
— 3	0	Reserved. Write to zero for future compatibility.
DE 2	0	**Delineation Error** Received unaligned /SC/ or /PD/ or undefined code-group.
UCS 1	0	**Unsolicited Control Symbol** An unsolicited acknowledge control symbol was received.
LTO 0	0	**Link Time-Out** An acknowledge or link-response control symbol is not received within the specified time-out interval.

16.6.31 Port 0–1 Error Rate Enable Command and Status Register (PnERECSR)

P0ERECSR Port 0–1 Error Rate Enable Command and Status Register Offset 0x00644
P1ERECSR Offset 0x00684

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—									CCS	AUA	PNA	UA	CRC	EM	—
TYPE	R/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**
	—										NOA	PE	—	DE	UCS	LTO
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnERECSR contains the bits that control when an error condition is allowed to increment the error rate counter in the Port 0–1 error rate threshold register and lock the Port 0–1 error capture registers.

Table 16-75. PnERECSR Field Descriptions

Bit	Reset	Description
— 31–23	0	Reserved. Write to zero for future compatibility.
CCS 22	0	**CRC Control Symbol Enable** Enable error counting of a corrupt control symbol.
AUA 21	0	**Acknowledge Control With Unexpected Acknowledge ID Enable** Enable error rate counting of an acknowledge control symbol with an unexpected acknowledge ID.
PNA 20	0	**Packet Not Accepted Enable** Enable error rate counting of packet-not-accepted acknowledge control symbols.
UA 19	0	**Unexpected Acknowledge ID Enable** Enable error rate counting of packets with an unexpected ackID value.
CRC 18	0	**Bad CRC Enable** Enable error rate counting of packets with a bad CRC value.
EM 17	0	**Exceed Maximum Enable** Enable error rate counting of packets that exceed the maximum allowed size (276 bytes).
— 16–6	0	Reserved. Write to zero for future compatibility.
NOA 5	0	**Not Outstanding Acknowledge** Enable error rate counting of ink-responses received with an acknowledge ID that is not outstanding.
PE 4	0	**Protocol Error** Enable error rate counting of protocol errors.
— 3	0	Reserved. Write to zero for future compatibility.
DE 2	0	**Delineation Errors** Enable error rate counting of delineation errors.

Table 16-75. PnERECSR Field Descriptions (Continued)

Bit	Reset	Description
UCS 1	0	**Unsolicited Control Symbol** Enable error rate counting of unexpected acknowledge control symbols.
LTO 0	0	**Link Time-Out** Enable error rate counting of an acknowledge or link-response control symbol not received within the specified time-out interval.

16.6.32 Port 0–1 Error Capture Attributes Command and Status Register (PnECACSR)

P0ECACSR Port 0–1 Error Capture Attributes Command and Status Register Offset 0x00648
P1ECACSR Offset 0x00688

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IT		—	ET					ECI15	ECI14	ECI13	ECI12	ECI11	ECI10	ECI9	ECI8
TYPE	R/W		R	R/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**
	ECI7	ECI6	ECI5	ECI4	ECI3	ECI2	ECI1	ECI0	—							CVI
TYPE	R/W								R							R/W
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnECACSR indicates the type of information contained in the Port 0–1 error capture registers. For multiple errors detected 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 detected by the port. Before the capture registers are read, software should verify that the PnECACSR[CVI] bit is set to ensure that the error is properly captured.

Table 16-76. PnECACSR Field Descriptions

Bit	Reset	Description	Settings
IT 31–30	0	**Information Type** 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 four words long. 01 Control symbol. Only the error capture register 0 is valid. 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 thee symbols.
— 29	0	Reserved. Write to zero for future compatibility.	

Table 16-76. PnECACSR Field Descriptions

Bit	Reset	Description	Settings
ET 28–24	0	**Error** The encoded value of the bit in the Port 0–1 error detect CSR that describes the error captured in the Port 0–1 error capture CSRs.	
ECI 23–8	0	**Extended Capture Information** ECI contains the control/data character signal corresponding to each byte of captured data.	ECI15] Associated with PnPCSECCSR0[31–24]. ECI14 Associated with PnPCSECCSR0[23–16]. ECI13 Associated with PnPCSECCSR00[15–8]. ECI12 Associated with PnPCSECCSR0[7–0]. ECI11 Associated with PnPECCSR1[31–24]. ECI10 Associated with PnPECCSR1[25–16]. ... ECI1 Associated with PnPECCSR3[15–8]. ECI0 Associated with PnPECCSR3[7–0].
— 7–1	0	Reserved. Write to zero for future compatibility.	
CVI 0		**Contain Valid Information** Hardware sets CVI to indicate that the packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.	

16.6.33 Port 0–1 Packet/Control Symbol Error Capture Command and Status Register (PnPCSECCSR)

P0PCSECCSR Port 0–1 Packet/Control Symbol Error Capture Command and Status Register Offset 0x0064C
P1PCSECCSR Offset 0x0068C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	C0															
TYPE	R/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**
	C0															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnPCSECCSR0 contains the first 4 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 detected by the port. Software should verify that the PnECACSR[CVI] bit is set before reading the capture registers to ensure that the error is properly captured.

Table 16-77. PnPCSECCSR0 Field Descriptions

Bit	Reset	Description
C0 31–0	0	**Capture 0** Control character and control symbol or bytes 0–3 of the packet header.

16.6.34 Port 0–1 Packet Error Capture Command and Status Register 1 (PnPECCSR1)

P0PECCSR1 Port 0–1 Packet Error Capture Command and Status Register 1 Offset 0x00650
P1PECCSR1 Offset 0x00690

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	C1															
TYPE	R/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**
	C1															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnPECCSR1 contains bytes 4–7 of the packet header. Undefined results occur if this register is written while actual physical layer errors are detected by the port. Software should verify that the PnECACSR[CVI] bit is set before reading the capture registers to ensure that the error is properly captured.

Table 16-78. PnPECCSR1 Field Descriptions

Bit	Reset	Description
C1 31–0	0	**Capture 1** Contains bytes 4–7 of the packet header.

16.6.35 Port 0–1 Packet Error Capture Command and Status Register 2 (PnPECCSR2)

P0PECCSR2 Port 0–1 Packet Error Capture Command and Status Register 2 Offset 0x00654
P1PECCSR2 Offset 0x00694

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	C2															
TYPE	R/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**
	C2															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnPECCSR2 contains bytes 8–11 of the packet header. Undefined results occur if this register is written while actual physical layer errors are detected by the port. Software should verify that the PnECACSR[CVI] bit is set before reading the capture registers to ensure that the error is properly captured.

Table 16-79. PECCSR2 Field Descriptions

Bit	Reset	Description
C2 31–0	0	**Capture 2** Bytes 8 to 11 of the packet header

16.6.36 Port 0–1 Packet Error Capture Command and Status Register 3 (PnPECCSR3)

P0PECCSR3 Port 0–1 Packet Error Capture Command and Status Register 3 Offset 0x00658
P1PECCSR3 Offset 0x00698

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	C3															
TYPE	R/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**
	C3															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnPECCSR3 contains bytes 12–15 of the packet header. Undefined results occur if this register is written while actual physical layer errors are detected by the port. Software should verify that the PnECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Table 16-80. PnPECCSR3 Field Descriptions

Bit	Reset	Description
C3 31–0	0	**Capture 3** Bytes 12–15 of the packet header.

16.6.37 Port 0–1 Error Rate Command and Status Register (PnERCSR)

P0ERCSR Port 0–1 Error Rate Command and Status Register Offset 0x00668
P1ERCSR Offset 0x006A8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ERB								—						ERR	
TYPE	R/W								R						R/W	
RESET	1	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**
	PER								ERC							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnERCSR is used with the Port0 Error Rate Threshold Command and Status Register (PnERTCSR) to monitor and control the reporting of transmission errors.

Table 16-81. PnERCSR Field Descriptions

Bit	Reset	Description	Setting
ERB 31–24	0b1000_0000	**Error Rate Bias** Provides 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.
— 23–18	0	Reserved. Write to zero for future compatibility.	
ERR 17–16	0	**Error Rate** Limits increments of the error rate counter above the failed threshold trigger. This counter never increments above 0xFF, even if the combined settings of ERR and the failed threshold trigger imply that it would.	0b00 Count only 2 errors above. 0b01 Count only 4 errors above. 0b10 Count only 16 error above. 0b11 Do not limit increments above the error rate count.
PER 15–8	0	**Peak Error Rate** Contains the peak value attained by the error rate counter.	
ERC 7–0	0	**Error Rate Counter** Counts the number of transmission errors detected by the port, decremented by the error rate bias, to create an indication of the link error rate. Software should not attempt to write to ERC a value higher than the failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).	

16.6.38 Port 0–1 Error Rate Threshold Command and Status Register (PnERTCSR)

P0ERTCSR Port 0–1 Error Rate Threshold Command and Status Offset 0x0066C
P1ERTCSR ‘Register Offset 0x006AC

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

PnERTCSR controls reporting of the link status to the system host.

Table 16-82. PnERTCSR Field Descriptions

Bit	Reset	Description	Settings
ERFTT 31–24	0xFF	**Error Rate Failed Threshold Trigger** Provides the threshold value for reporting an error condition due to a possibly broken link. The PnESCSR[OFE] bit is set if PnERCSR[ERC] meets or exceeds the ERFTT value (that is, PnERCSR[ERC] ≥ ERFTT).	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.
ERDTT 23–16	0xFF	**Error Rate Degraded Threshold Trigger** Provides the threshold value for reporting an error condition due to a degrading link. The PnESCSR[ODE] bit is set if PnERCSR[ERC] meets or exceeds the ERDTT value (that is, PnERCSR[ERC] ≥ ERFTT).	0x00 Disable the error rate degraded 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.
— 15–0	0	Reserved. Write to zero for future compatibility.	

16.6.39 Logical Layer Configuration Register (LLCR)

LLCR Logical Layer Configuration Register Offset 0x10004

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		ECRAB	—												
TYPE	R/W			R												
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**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LLCR contains general port-common logical layer mode enables.

Table 16-83. LLCR Field Descriptions

Bit	Reset	Description
— 31–30	0	Reserved. Write to zero for future compatibility.
ECRAB 29	0	**External Configuration Register Access Block** Blocks all maintenance requests and accesses to LCSBA1CSR. Reads return all 0s, and writes are ignored (both return a done response). When ECRAB is cleared, any external RapidIO device can access the registers.
— 28–0	0	Reserved. Write to zero for future compatibility.

16.6.40 Error/Port-Write Status Register (EPWISR)

EPWISR Error/Port-Write Interrupt Status Register Offset 0x10010

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PINT0	PINT1	—													
TYPE	R															
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**
	—														MU	PW
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EPWISR contains status bits of the interrupts generated by the port or the message unit for physical or logical/transport layer errors or inbound port-writes. Because errors from the port are reported to the SC3850 core with one interrupt signal, this register provides the core with quick access to where the error occurred. This register is read-only and stored in the port and the message unit as a logically equivalent copy.

Table 16-84. EPWISR Field Descriptions

Bit	Reset	Description
PINT0 31	0	**Physical or Logical/Transport Error Interrupt Port 0** Indicates a physical or logical transport error interrupt was generated by port 0.
PINT1 30	0	**Physical or Logical/Transport Error Interrupt Port 1** Indicates a physical or logical transport error interrupt was generated by port 1
— 29–2	0	Reserved. Can be used to indicate errors on more ports if they exist.
MU 1	0	**Message Unit Logical/Transport Layer Error Interrupt** Indicates a logical/transport layer error interrupt was generated by the message unit.
PW 0	0	**Port Write** Indicates an inbound port-write was received.

16.6.41 Logical Retry Error Threshold Configuration Register (LRETCR)

LRETCR Logical Retry Error Threshold Configuration Register Offset 0x10020

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R/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**
	—								RET							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

LRETCR contains the retry error threshold for the logical layer. When the number of consecutive logical retries for a given packet is greater than this value, an error interrupt is generated. Notice that the number of retries must be greater than the threshold value, which is not the case for other registers that define a retry threshold.

Table 16-85. LRETCR Field Descriptions

Bit	Reset	Description	
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0xFF	**Retry Error Threshold** The threshold value for the number of consecutive logical retries received for a given packet that causes the RAPIDIO ENDPOINT to report an error condition.	0x00 Disable the retry threshold. 0x01 Set the error reporting threshold to 1. ... 0xFF Set the error reporting threshold to 255.

16.6.42 Physical Retry Error Threshold Configuration Register (PRETCR)

PRETCR Physical Retry Error Threshold Configuration Register Offset 0x10080

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—								RET							
TYPE	R								R/W							
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

PRETCR contains the retry error threshold for the physical layer. When the number of consecutive acknowledge-retries is greater than or equal to this value, an error interrupt is generated.

Table 16-86. PRETCR Field Descriptions

Bit	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0xFF	**Retry Error Threshold** The threshold value for the number of consecutive acknowledge-retries received that cause the RAPIDIO ENDPOINT to report an error condition.	0x00 Disable the retry threshold. 0x01 Set the error reporting threshold to 1. ... 0xFF Set the error reporting threshold to 255.

16.6.43 Port 0–1 Alternate Device ID Command and Status Register (PnADIDCSR)

P0ADIDCSR Port 0–1 Alternate Device ID Command and Status Register Offset 0x10100
P1ADIDCSR Offset 0x10180

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ADE	—							ADID							
TYPE	R/W	R							R/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**
	LADID															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnADIDCSR contains an alternate device ID. This register should be enabled before the initiator enabled bit of the PGCCSR is set (see **Table 16-62, *PGCCSR Field Descriptions,*** on page 16-150). Therefore, when the PGCCSR bit is enabled, all other devices in the RapidIO system (including switches) send and receive packets from the device ID in PnADIDCSR instead of the device ID in BDIDCSR. When the alternate deviceID is enabled, inbound RapidIO endpoint accepts only packets sent with the device ID in PnADIDCSR or the deviceID in BDIDCSR. An exception is Accept All mode, in which the inbound RapidIO Endpoint accept packets using the same common transport system. The outbound RapidIO endpoint generates requests using only the device ID in PnADIDCSR. It generates responses with the deviceID in the original request packet (either from PnADIDCSR or BDIDCSR). The selection between a large or small transport system is done during the power-up sequence by using the CTLS bit in the RCW.

Table 16-87. PnADIDCSR Field Descriptions

Bit	Reset	Description
ADE 31	0	**Alternate Device ID Enable** Causes the port to use the device ID specified in this register instead of the device ID specified in BDIDCSR.
— 30–24	0	Reserved. Write to zero for future compatibility.
ADID 23–16	0	**Alternate Device ID** Alternate device ID in a small transport system.
LADID 15–0	0	**Large Alternate Device ID** Alternate device ID for the device in a large transport system.

16.6.44 Port 0–1 Pass-Through Accept-All Configuration Register (PnPTAACR)

P0PTAACR Port 0–1 Pass-Through Accept-All Configuration Register Offset 0x10120
P1PTAACR Offset 0x101A0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PTPN				—											
TYPE	R/W															
RESET	x	x	x	x	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**
	—														PTE	AA
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x

PnPTAACR contains information on accept-all mode.

Table 16-88. PnPTAACR Field Descriptions

Bit	Reset	Description	Settings
PTPN 31–28	Port 0 = 1 Port 1 = 0	**Pass-Through RapidIO Port Number** The RapidIO port to which to send pass-through packets. The SRIO unit in the HSSI uses this value to map the RapidIO port number (0 or 1) to the OCN port number (0 or 4).	
— 27–2	0	Reserved. Write to zero for future compatibility.	
PTE 1	RCWHR[RPT]	**Pass-Through Mode Enable** Enables/disables pass-through mode. **Note:** AA has priority over PTE; when AA is set, the value of PTE is ignored.	0 Packets whose target ID does not match the device ID are not sent out to another SRIO port. An error occurs if the AA bit is not set (to select accept all mode). 1 Packet whose target ID does not match the device ID (base or alternate, if enabled) or whose transport size does not match the device transport size (defined by the common transport large system bit in the high part of the Reset Configuration Word (RCWH[CTLS]).is sent out through the OCN to another SRIO port designated by the value of the PTPN field.
AA 0	Port 0 = RCWHR[R1A] Port 1 = RCWHR[R2A]	**Accept All** Specifies whether packet acceptance is based on a target ID. When this bit is set, the tt field value must be consistent with the common transport system specified by the CTLS bit of the processing element features CAR.	0 Normal RapidIO acceptance based on target ID. 1 All packets are accepted without checking the target ID.

16.6.45 Port 0–1 Logical Outbound Packet Time-to-Live Configuration Register (PnLOPTTLCR)

P0LOPTTLCR Port 0–1 Logical Outbound Packet Time-to-Live Configuration Register Offset 0x10124
P1LOPTTLCR Offset 0x101A4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TV															
TYPE	R/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**
	TV								—							
TYPE	R/W								R							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Port 0–1 logical outbound packet time-to-live configuration register (PnLOPTTLCR) 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 will be 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). The reset value is the maximum time-out interval and represents between 3 and 5 seconds. When the packet time-to-live counter expires, PnPCR[OBDEN] is automatically set. PnPCR[OBDEN] must be cleared by software.

The time-out increment unit is a function of the OCN bus clock frequency (the HSSI clock frequency) and the value given in the RapidIO prescaler field (CLK_GPR0[RPTE]) and is within the range of 253 ns (±7 ns). The exact formula to calculate the value is:

```
Timer Resolution = 2 * (RPTE +1)/ocn_clk_freq
where, ocn_clk_freq = HSSI clock frequency
```

For example, if `ocn_clk_freq` = 333 MHz (RCWR[MODCK] = 0) and CLK_GPR0[RPTE] = 41, the timer resolution is 252 ns.

Table 16-89. PnLOPTTLCR Field Descriptions

Bit	Reset	Description
TV 31–8	0	**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.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.46 Port 0–1 Implementation Error Command and Status Register (PnIECSR)

P0IECSR Port 0–1 Implementation Error Command and Status Register — Offset 0x10130
P1IECSR — Offset 0x101B0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RETE	—														
TYPE	W1C	R														
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**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnIECSR contains status bits that are asserted when an implementation-defined error occurs.

Table 16-90. PnIECSR Field Descriptions

Bit	Reset	Description
RETE 31	0	**Retry Error Threshold Exceeded** Set when the number of consecutive retries reaches the retry error threshold in the Physical Retry Error Threshold Configuration Register (PRETCR). RETE is cleared by writing a value of 1 to it. This bit is set again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.
— 30–0	0	Reserved. Write to zero for future compatibility.

16.6.47 Port 0–1 Serial Link Command and Status Register (PnSLCSR)

P0SLCSR Port 0–1 Serial Link Command and Status Register Offset 0x10158
P1SLCSR Offset 0x101D8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LS0	LS1	LS2	LS3	—				LA	—						
TYPE	W1C	W1C	W1C	W1C	R				W1C	R						
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**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnSLCSR contains status of the of the serial physical link.

Table 16-91. PnSLCSR Field Descriptions

Bit	Reset	Description
LS0 31	0	**Lane Sync Achieved for Lane 0** Write with 1 to clear
LS1 30	0	**Lane Sync Achieved for Lane 1** Write with 1 to clear
LS2 29	0	**Lane Sync Achieved for Lane 2** Write with 1 to clear
LS3 28	0	**Lane Sync Achieved for Lane 3** Write with 1 to clear
— 27–24	0	Reserved. Write to zero for future compatibility.
LA 23	0	**Lane Alignment Achieved** Write with 1 to clear.
— 22–0	0	Reserved. Write to zero for future compatibility.

16.6.48 Port 0–1 Serial Link Error Injection Configuration Register (PnSLEICR)

P0SLEICR Port 0–1 Serial Link Error Injection Configuration Register Offset 0x10160
P1SLEICR Offset 0x101E0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EIC					—							EIR			
TYPE	R/W					R							R/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**
	EIR															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnSLEICR controls the injection of bit errors into the transmit bit stream and is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is non-zero, error injection is enabled and, at pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

Table 16-92. PnSLEICR Field Descriptions

Bit	Reset	Description	Settings
EIC 31–27	0	**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 lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes All other values reserved.
— 26–20	0	Reserved. Write to zero for future compatibility.	
EIR 19–0	0	**Error Injection Range** The value of EIR × 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 indicates a maximum delay of 32 character times. The value within this register should be right-justified.	

16.6.49 IP Block Revision Register 1 (IPBRR1)

IPBRR1 IP Block Revision Register 1 Offset 0x10BF8

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

IPBRR1 tracks changes and revisions of the RapidIO endpoint.

Table 16-93. IPBRR1 Field Descriptions

Bit	Reset	Description
IPID 31–16	0x01C0	IP block ID = 0x01C0
IPMJ 15–8	0x01	Major revision of the IP block = 0x01
IPMN 7–0	0x00	Minor revision of the IP block = 0x02

16.6.50 IP Block Revision Register 2 (IPBRR2)

IPBRR2 IP Block Revision Register 2 Offset 0x10BFC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								IPINT							
TYPE	R															
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**
	—								IPCFG							
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

IPBRR2 track changes and revisions of the RapidIO endpoint.

Table 16-94. IPBRR2 Field Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
IPINT 23–16	0	IP block Integration options = 0x0.
— 15–8	0	Reserved. Write to zero for future compatibility.
IPCFG 7–0	0x01	IP Block Configuration Options = 0x01.

16.6.51 Port 0–1 RapidIO Outbound Window Translation Address Registers x (PnROWTARx)

P0ROWTAR[0–8] Port 0–1 RapidIO Outbound Window Translation Address Registers 0–8 Offset 0x10C00 + x*0x20
P1ROWTAR[0–8] Offset 0x10E00 + x*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LTGTID		TREXAD										TRAD			
TYPE	R/W															
Reset																
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1–8	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**
	TRAD															
TYPE	R/W															
Reset																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1–8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnROWTARx points to the starting addresses in the RapidIO 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.

Table 16-95. PnROWTAR*x* Field Descriptions

Bit	Reset	Description
LTGTID 31–30	0	**Large Transport System Target Address** LTGTID corresponds to bits 6–7 of the targetID for a large transport system. This field is valid only if the PEFCAR[CTLS] bit is set (see **Section 16.6.5**, *Processing Element Features Capability Register (PEFCAR),* on page 16-135). Bits 0–5 of the target ID are specified in the PnROWTEARX (see **Section 16.6.52**, *Port 0–1 RapidIO Outbound Window Translation Extended Address Registers x (PnROWTEARx),* on page 16-186
TREXAD 29–20	Reg. 0: 0x008 Reg. 1–8: 0	**Translation Extended Address** TREXAD[0–7] corresponds 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] correspond to bits 0–1 of a 34-bit RapidIO translation address. For maintenance transactions and default window 0, TREXAD[8–9] are reserved.
TRAD 19–0	0	**Translation Address** System address that represents the starting point of the outbound translated address. The translation address must be aligned based on the field size. 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 reserved for default window 0.

16.6.52 Port 0–1 RapidIO Outbound Window Translation Extended Address Registers x (PnROWTEARx)

P0ROWTEAR[0–8]	Port 0–1 RapidIO Outbound Window Translation Extended Address Registers 0–8	Offset 0x10C04 + x*0x20
P1ROWTEAR[0–8]		Offset 0x10E04 + x*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R/W															
Reset																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1–8	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
	—										LTGTID					
TYPE	R/W															
Reset																
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1–8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnROWTEARx points to the starting addresses in the RapidIO 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.

Table 16-96. PnROWTEAR*x* Field Descriptions

Bit	Reset	Description
— 31–6	0	Reserved. Write to zero for future compatibility.
LTGTID 5–0	Reg. 0: 111111 Reg. 1–8: 000000	**Large Transport System Target ID** Corresponds to bits 0–5 of the target ID for a large transport system. This field is valid only if the PEFCAR[CTLS] bit is set (see **Section 16.6.5**, *Processing Element Features Capability Register (PEFCAR),* on page 16-135). Bits 6–7 of the target ID are specified in the corresponding PnROWTARx.

16.6.53 Port 0–1 RapidIO Outbound Window Base Address Registers x (PnROWBARx)

P0ROWBAR[1–8] Port 0–1 RapidIO Outbound Offset 0x10C08 + x*0x20
P1ROWBAR[1–8] Window Base Address Registers 1–8 Offset 0x10E08 + x*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								BEXAD				BADD			
TYPE	R								R/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**
	BADD															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnROWBARx selects the base address for the windows that are translated to an alternate target address space. Addresses for outbound transactions are compared with the addresses of these windows. If such a transaction does not fall within one of these spaces, it is forwarded to the interior of the device using the default window. For information on transactions that cross more than one window, see **Section 16.2.5.4.2**, *Window Boundary Crossing Errors*, on page 16-44.

Table 16-97. PnROWBAR*x* Field Descriptions

Bit	Reset	Description
— 31–24	0	Reserved. Write to zero for future compatibility.
BEXAD 23–20	0	**Base Extended Address** Bits 0–3 of the 36-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.
BADD 19–0	0	**Base Address** A system address that is the starting-point for the outbound translation window. The window must be aligned on the basis of the size selected in the window size bits. This corresponds to bits 4–23 of the 36-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.

16.6.54 Port 0–1 RapidIO Outbound Window Attributes Registers x (PnROWARx)

P0ROWARx Port 0–1 RapidIO Outbound Offset 0x10C10 + x*0x20
P1ROWARx Window Attributes Registers 0–8 Offset 0x10E10 + x*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	—			TFLOWLV		PCI	—	NSEG		NSSEG		RDTYP			
Type	R/W	R			R/W			R	R/W							
Reset:																
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1–8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WRTYP				—						SIZE					
Type	R/W				R						R/W (R for default window 0)					
Reset:																
0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1
1–8	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

The RapidIO outbound window attributes registers define the window sizes to translate and other attributes of the translations. The largest window size allowed is 64 GB. For a segmented window, these attributes are used for segment 0. The PCI window bit applies for all segments.

Table 16-98. PnROWAR*x* Field Descriptions

Bit	Reset	Description	Settings
EN 31	Reg. 0: 1 Reg. 1–8: 0	**Enable Address Translation** For default window 0 only, this bit is set to 1 and read-only.	0 Window disabled. 1 Window enabled.
— 30–28	000	Reserved. Write to zero for future compatibility.	
TFLOWLV 27–26	**00**	**Transaction Flow Level** Selects the transaction flow priority level. This field must be cleared (00) if the PCI bit is set. Also, the RapidIO priority given by this field must always be greater than or equal to the internal priority or a deadlock can occur. Normally, the internal priority of all packets is 0.	00 Lowest priority transaction request flow. 01 Medium priority transaction request flow. 10 Highest priority transaction request flow. 11 Reserved.
PCI 25	**0**	**PCI Window** If set, this window follows PCI ordering rules as defined in the RapidIO Inter-operability specification. The TFLOWLV field must be 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.	0 Non-PCI window rules. 1 PCI window rules.
— 24	0	Reserved. Write to zero for future compatibility.	
NSEG 23–22	00	**Number of Segments** Number of segments for this window. **Note:** This field is reserved for default window 0.	00 One segment (normal) 01 Two segments (half-size aliasing window) 10 Four segments (quarter-size aliasing window) 11 Reserved.

Table 16-98. PnROWAR*x* Field Descriptions (Continued)

Bit	Reset	Description	Settings
NSSEG 21–20	00	**Number of Subsegments per Segment** Defines the number of segments to use with each segment. **Notes:** **1.** This field is reserved for default window 0. **2.** This field is valid only if NSEG = 1 or 2.	00 One target deviceID per segment 01 Two target deviceIDs per segment. 10 Four target deviceIDs per segment. 11 Eight target deviceIDs per segment.
RDTYP 19–16	0100	**Read Type** Transaction type to run on the RapidIO interface if the access is a read.	0100 Read. 0111 Maintenance Read. All other values are reserved.
WRTYP 15–12	0100	**Write Type** Transaction type to run on the RapidIO interface if access is a write. A write-requiring-response sent from an internal source must generate a write-requiring-response to the RapidIO interface. Therefore, if an internal write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates an NWRITE_R instead.	0011 SWRITE. 0100 NWRITE. 0101 NWRITE_R. 0111 Maintenance Write All other values are reserved.
— 11–6	000000	Reserved. Write to zero for future compatibility.	
SIZE 5–0	000011	**Window Size** Outbound translation window size N, which is the encoded 2^(N+1) byte window size. The smallest window size is 4 Kbyte. **Note:** This field is read-only for default window 0.	000000 Reserved. ... 001011 4 KB window size. 001100 8 KB window size. ... 011111 4 GB window size. 100000 8 GB window size. 100001 16 GB window size. 100010 32 GB window size. 100011 64 GB window size. 100100 Reserved ... 111111 Reserved.

16.6.55 Port 0–1 RapidIO Outbound Window Segment 1–3 Registers 1–8 (PnROWSxRy)

Port 0–1 RapidIO Outbound Window Segment 1–3 Registers 1–8

P0ROWS[1–3]R[1–8] Offset 0x10C34 + (x–1)*0x4 + (y–1)*0x20

P1ROWS[1–3]R[1–8] Offset 0x10E34 + (x–1)*0x4 + (y–1)*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				TFLOWLV		—		RDTYP				WRTYP			
TYPE	R				R/W		R		R/W							
RESET	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	—								SGTGTDID							
TYPE	R								R/W							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnROWSxRy define the attributes and target device ID to use for a transaction that hits in the segment rather than the primary attributes and target deviceID.There is a segment register for each segment except segment 0.

Table 16-99. PnROWSxRn Field Descriptions

Bit	Description	Settings
— 31–28	Reserved. Write to zero for future compatibility.	
TFLOWLY 27–26	**Transaction Flow Level** Selects the transaction flow priority level. **Note:** This field must be set to 00 if the PCI bit is set.	00 Lowest priority transaction request flow. 01 Next highest priority transaction request flow. 10 Highest priority transaction request flow. 11 Reserved.
— 25–24	Reserved. Write to zero for future compatibility.	
RDTYP 23–20	**Read Type** Transaction type to run on the RapidIO interface if the access is a read.	0100 NREAD. 0111 Maintenance read. All other values are reserved.
WRTYP 19–16	**Write Type** Transaction type to run on the RapidIO interface if access is a write.	0011 SWRITE. 0100 NWRITE. 0101 NWRITE_R. 0111 Maintenance Write. All other values are reserved.
— 15–8	Reserved. Write to zero for future compatibility.	
SGTGTDID 7–0	**Segment Target DeviceID** Stores the Target DeviceID as follows: • SGTGTDID[7–3]: Bits 0–4 for small transport or bits 8–12 for large transport. • SGTGTDID2: Bit 5 for small transport or bit 13 for large transport; reserved for 8 target subsegments. • SBTGTDID1: Bit 6 for small transport or bit 14 for large transport; reserved for 8 or 4 target subsegments. • SBTGTDID0: Bit 7 for small transport or bit 15 for large transport; reserved for 8, 4, or 2 target subsegments.	

16.6.56 Port 0–1 RapidIO Inbound Window Translation Address Registers x (PnRIWTARx)

P0RIWTAR[0–4] Port 0–1 RapidIO Inbound Window Translation Address Registers 0–4 — Offset 0x10D60 + (4–x)*0x20

P1RIWTAR[0–4] — Offset 0x10F60 + (4–x)*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												TRAD			
TYPE	R/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**
	TRAD															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnRIWTARx points to the starting addresses in the RapidIO 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.

Table 16-100. PnRIWTAR*x* Field Descriptions

Bit	Reset	Description
— 31–20	0	Reserved. Write to zero for future compatibility.
TRAD 19–0	0	**Translation Address** Target address to indicate the starting point of the inbound translated address. The translation address must be aligned on the basis of the size field. TRAD is reserved for default window 0.

16.6.57 Port 0–1 RapidIO Inbound Window Base Address Registers x (PnRIWBARx)

P0RIWBAR[1–4] Port 0–1 RapidIO Inbound Window Base Address Registers 1–4 — Offset 0x10D68 + (4–x)*0x20

P1RIWBAR[1–4] — Offset 0x10F68 + (4–x)*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—										BEXAD		BADD			
TYPE	R										R/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**
	BADD															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PnRIWBARx selects the base address for the windows that are translated to an alternate target address space. Addresses for inbound transactions are compared with the addresses of these windows. If such a transaction does not fall within one of these spaces, it is forwarded to the interior of the device using the default window. For information on transactions that cross more than one window, see **Section 16.2.5.4.2**, *Window Boundary Crossing Errors*, on page 16-44.

Table 16-101. PnRIWBAR*x* Field Descriptions

Bit	Reset	Description
— 31–22	0	Reserved. Write to zero for future compatibility.
BEXAD 21–20	0	**Base Extended Address** Bits 0–1 of the 34-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.
BADD 19–0	0	**Base Address** A system address that is the starting-point for the inbound translation window. The window must be aligned on the basis of the size selected in the window size bits. This corresponds to bits 2–21 of the 34-bit RapidIO base address. **Note:** Bit 0 is the most significant bit.

16.6.58 Port 0–1 RapidIO Inbound Window Attributes Registers x (PnRIWARx)

P0RIWARx Port 0–1 RapidIO Inbound Window Attributes Registers 0–4 Offset 0x10D70 + (4–x)*0x20
P1RIWARx Offset 0x10F70 + (4–x)*0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	PW	—						TGINT				RDTYP			
TYPE	R/W		R						R/W							
RESET	0000_0000_0000_0100 (PnRIWAR 1–4) 1000_0000_0000_0100 (PnRIWAR 0)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WRTYP				—						SIZE					
TYPE	R/W				R						R/W (R for default window 0)					
RESET	0100_0000_0010_0001 (PnRIWAR 0–4)															

The port0 RapidIO inbound window attributes registers define the window sizes to translate and other attributes of the translations. The largest window size allowed is 16 GB. The RDTYP and WRTYP fields are used for attributes on the request, but they do not change the type of the transaction. In other words, PnRIWAR*x* does not modify whether the request requires a response or not. This type of attribute remains unchanged on the request as it is translated through the ATMU.

Table 16-102. PnRIWAR*x* Field Descriptions

Bit	Description	Settings
EN 31	**Enable Address Translation** Set to 1 and read-only for default window 0.	
PW 30	**Protected Window** Indicates that this window is protected. Writes requiring a response and reads to this window generate an error response. Writes not requiring a response are silently discarded.	
— 29–24	Reserved. Write to zero for future compatibility.	
TGINT 23–20	**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.	*For Port 0:* 0000 OCN to MBus 0 0001 OCN to MBus 1 0010 SRIO Port 0 0011 SRIO Port 1 0100–1111 Reserved. *For Port 1:* 0000 OCN to MBus 1 0001 OCN to MBus 0 0010 SRIO Port 0 0011 SRIO Port 1 0100–1111 Reserved.

Table 16-102. PnRIWAR*x* Field Descriptions (Continued)

Bit	Description	Settings
RDTYP 19–16	**Read Type** Transaction type to run on the local memory if the access is a read.	0000 Reserved. ... 0011 Reserved. 0100 Read. 0101 Reserved. ... 1111 Reserved.
WRTYP 15–12	Transaction type to run on local memory if access is a write.	0000 Reserved. ... 0011 Reserved. 0100 Write. 0101 Reserved. ... 1111 Reserved.
— 11–6	Reserved. Write to zero for future compatibility.	
SIZE 5–0	**Window Size** Inbound translation window size N, which is the encoded 2^(N+1) byte window size. The smallest window size is 4 KB. This field is read-only for default window 0.	000000 Reserved. ... 001011 4 KB window size. 001100 8 KB window size. ... 011111 4 GB window size. 100000 8 GB window size. 100001 16 GB window size. 100010 Reserved. ... 111111 Reserved.

16.6.59 Outbound Message x Mode Registers (OMxMR)

OM[0–1]MR Outbound Message 0–1 Mode Registers Offset 0x13000 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—				SCTL			—								
TYPE	R				R/W			R								
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**
	CIRQ_SIZ				—		QOIE	QFIE	—	QEIE	EIE	—		MUTM	MUI	MUS
TYPE	R/W				R		R/W		R	R/W		R		R/W		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxMR allows software to start a message operation and to control various message operation characteristics.

Table 16-103. OM*x*MR Field Descriptions

Bits	Reset	Description	
— 31–28	0	Reserved. Write to zero for future compatibility.	
SCTL 27–25	0	**Service Control** Determines the number of descriptors to process before the next queue is serviced. Note that if one queue has SCTL set to fixed priority, all SCTL values in other queues are ignored. For example, of the two outbound message units and the service control value for unit 1 is set to 0b000, a fixed priority is set and unit 0 handles the highest priority results. If a queue is in direct mode, only one message operation is serviced before the next queue is serviced. For proper operation, this field should be modified only when the outbound message controller is not enabled. The value of this field cannot be changed unless both units are disabled.	000 Fixed priority based on outbound message unit number. 001 1 descriptor. 010 2 descriptors. 011 4 descriptors. 100 8 descriptors. 101 16 descriptors. 110 32 descriptors. 111 64 descriptors.
— 24–16	0	Reserved. Write to zero for future compatibility.	
CIRQ_SIZ 15–12	0	**Circular Descriptor Queue Size** Determines the number of descriptors that can be placed on the circular queue without overflow. For proper operation, this field should be modified only when the outbound message controller is not enabled	0000 2. 0001 4. 0010 8. 0011 16. 0100 32. 0101 64. 0110 128. 0111 256. 1000 512. 1001 1024. 1010 2048. 1011— 1111 Reserved.
— 11–10	0	Reserved. Write to zero for future compatibility.	
QOIE 9	0	**Queue Overflow Interrupt Enable** Enables an interrupt when a queue overflow is detected. That is, the enqueue and dequeue pointers are no longer equal after the processor increments the enqueue pointer while the queue is full. This bit is applicable only in chaining mode. No queue overflow interrupt is generated if this bit is cleared. If this bit is not set and the queue overflows, the result is undefined.	
QFIE 8	0	**Queue Full Interrupt Enable** Enables an interrupt when the queue transitions to full. That is, the enqueue and dequeue pointers are equal after the processor increments the enqueue pointer. No QF interrupt is generated if this bit is cleared. If this bit is set and OM*x*SR[QF] is set, OM*x*SR[QFI] becomes set.	
— 7	0	Reserved. Write to zero for future compatibility.	

Table 16-103. OM*x*MR Field Descriptions (Continued)

Bits	Reset	Description	
QEIE 6	0	**Queue Empty Interrupt Enable** Enables an interrupt at the completion of all outstanding message operations. That is, the enqueue and dequeue pointers are equal after an increment by the message unit controller. No Queue Empty interrupt is generated if this bit is cleared. For proper operation, this field should be modified only when the outbound message controller is not enabled	0 No interrupt. 1 Queue empty interrupt.
EIE 5	0	**Error Interrupt Enable** Enables a port-write/error interrupt when a transfer error (OM*x*SR[TE]), a message error response (OM*x*SR[MER]), a packet response time-out (OM*x*SR[PRT]), or a retry threshold event exceeded (OM*x*SR[RETE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.	0 No port-write/error interrupt. 1 Generate a port-write/error interrupt.
— 4–3	0	Reserved. Write to zero for future compatibility.	
MUTM 2	0	**Message unit Transfer Mode** Puts the message unit into direct mode so that software is responsible for placing all the required parameters into registers to start the message transmission. Clearing this bit configures the message unit in chaining mode.	0 Chaining mode. 1 Direct mode.
MUI 1	0	**Message Unit Increment** Software sets this bit after writing a descriptor to memory. Hardware then increments the OM*x*DQEPAR and clears this bit. MUI always reads as 0 when MUS is set.	
MUS 0	0	**Message Unit Start** In Direct mode, a 0 to 1 transition when the message unit is not busy (MUB bit is 0) starts the message unit. A 1 to 0 transition has no effect. If this bit is set in Chaining mode, the message unit starts when the enqueue and dequeue pointers are not equal.	

16.6.60 Outbound Message x Status Registers (OMxSR)

OM[0–1]SR Outbound Message 0–1 Status Registers Offset 0x13004 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—											QF	—			
TYPE	R															
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**
	—			MER	RETE	PRT	—		TE	—	QOI	QFI	—	MUB	EOMI	QEI
TYPE	R			W1C	W1C	W1C	R		W1C	R	W1C	W1C	R		W1C	W1C
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxSR reports various message unit conditions during and after a message operation. Writing a 1 to the corresponding set bit clears the bit.

Table 16-104. OM*x*SR Field Descriptions

Bits	Reset	Description
— 31–21	0	Reserved. Write to zero for future compatibility.
QF 20	0	**Queue Full** If the queue becomes full, this bit is set. Read only.
— 19–13	0	Reserved. Write to zero for future compatibility.
MER 12	0	**Message Error Response** Set when an ERROR response is received from the message target. The error response received field indicates the value of the error response status bits when an error response is received. This bit is cleared by writing a value of 1 to it.
RETE 11	0	**Retry Error Threshold Exceeded** Set when the message unit is unable to complete a message operation because the retry error threshold value is exceeded due to a RapidIO retry response. This bit is cleared by writing a value of 1 to it.
PRT 10	0	**Packet Response Time-Out** Set when the message unit has been unable to complete a message operation and a packet response time-out occurred. This bit is cleared by writing a 1.
— 9–8	0	Reserved. Write to zero for future compatibility.
TE 7	0	**Transaction Error** Set when an internal error condition occurs during the message operation. This bit is cleared by writing a value of 1 to it. For proper operation, this field should be modified only when the outbound message controller is not enabled
— 6	0	Reserved. Write to zero for future compatibility.
QOI 5	0	**Queue Overflow Interrupt** Set when a queue overflow condition is detected. This bit is cleared by writing a value of 1 to it. QOI is applicable only to chaining mode.
QFI 4	0	**Queue Full Interrupt** If the queue becomes full and the QFIE bit in the Mode Register is set, this bit is set and an interrupt is generated. This bit is cleared by writing a value of 1 to it.
— 3	0	Reserved. Write to zero for future compatibility.
MUB 2	0	**Message Unit Busy** Indicates that a message operation is currently in progress. This bit is cleared when an error occurs or the message operation completes. Read only.
EOMI 1	0	**End-Of-Message Interrupt** When the message operation completes and the EOMIE bit in the Destination Attributes Register is set, EOMI is set and an interrupt is generated. This bit is cleared by writing a value of 1 to it.
QEI 0	0	**Queue Empty Interrupt** When the last message operation in the outbound descriptor queue is finished and the QEIE bit in the Mode Register is set, this bit is set and an interrupt is generated. Otherwise, no interrupt is generated. This bit is cleared by writing a value of 1 to it.

16.6.61 Outbound Message x Descriptor Queue Dequeue Pointer Address Registers (DMxDQDPAR)

OM[0–1]DQDPAR Outbound Message 0–1 Descriptor Queue Dequeue Pointer Address Registers — Offset 0x1300C + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DQDPA															
TYPE	R/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**
	DQDPA											—				
TYPE	R/W											R				
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxDQDPAR contain the address of the first descriptor in memory to be processed. Software must initialize this register to point to the first descriptor in memory. After the descriptor is processed, the message unit controller increments the outbound message descriptor queue dequeue pointer address in OM*x*DQDPAR to point to the next descriptor. If the outbound message descriptor queue enqueue pointer and the outbound message descriptor queue dequeue pointer are not equal (indicating that the queue is not empty), the message unit controller reads the next descriptor from memory for processing. If the enqueue and dequeue pointers are equal after the message unit controller increments the dequeue pointer, the queue is empty and the message unit halts until the processor increments the enqueue pointer. Incrementing the pointer indicates that a new descriptor was added to the queue and is ready for processing. If the queue becomes empty and OM*x*MR[QEIE] is set, OM*x*SR[QEI] is set and an interrupt is generated.

When software initializes these registers, they must be aligned on a boundary equal to the number of queue entries × 32 bytes (the size of each queue descriptor).For example, if there are eight entries in the queue, the register must be 256-byte aligned. The number of queue entries is set in OMnMR[CIRQ_SIZ].

Table 16-105. OM*x*DQDPAR Field Descriptions

Bits	Reset	Description
DQDPA 31–5	0	**Descriptor Dequeue Pointer Address** Contains the address of the first descriptor in memory to process. The descriptor must be aligned to a 32-byte boundary. For proper operation, this field should be modified only when the outbound message controller is not enabled.
— 4–0	0	Reserved. Write to zero for future compatibility.

16.6.62 Outbound Message x Source Address Registers (OMxSAR)

OM[0–1]SAR Outbound Message 0–1 Source Address Registers — Offset 0x13014 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SAD															
TYPE	R/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**
	SAD													—		
TYPE	R/W													R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxSAR indicates the address from which the message unit controller is to read data. Software must ensure that this is a valid local memory address. The source address must be aligned to a double-word boundary, so the least significant three bits are reserved.

Table 16-106. OM*x*SAR Field Descriptions

Bits	Reset	Description
SAD 31–3	0	**Source Address** The source address of the message operation. The contents are updated after every memory read operation. For proper operation, this field should be modified only when the outbound message controller is not enabled
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.63 Outbound Message x Destination Port Register (OMxDPR)

OM[0–1]DPR Outbound Message 0–1 Destination Port Registers — Offset 0x13018 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EDGTROUTE								DTGTROUTE							
TYPE	R/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**
	—														MAILB	
TYPE	R														R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxDPR indicates the RapidIO destination ID and mailbox to which the message unit controller is to send data. The software must ensure that this is a valid port in the receiving device.

Table 16-107. OM*x*DPR Field Descriptions

Bits	Reset	Description
EDGTROUTE 31–24	0	**Extended Destination Target Route** Most significant byte of a 16-bit target route when activated in Large Transport mode. Reserved when operated in Small Transport mode. For proper operation, this field should be modified only when the outbound message controller is not enabled
DTGTROUTE 23–16	0	**Destination Target Route** Contains the target route field of the transaction (device ID of the target). This value is overridden by the multicast group and list if Multicast mode is enabled. When an error occurs while Multicast mode is enabled, this field is loaded with the destination of the failed operation. For proper operation, this field should be modified only when the outbound message controller is not enabled
— 15–2	0	Reserved. Write to zero for future compatibility.
MAILB 1–0	0	**Value for MBOX Field in MESSAGE Packet** For proper operation, this field should be modified only when the outbound message controller is not enabled

16.6.64 Outbound Message x Destination Attributes Register (OMxDATR)

OM[0–1]DATR Outbound Message 0–1 Destination Attributes Registers — Offset 0x1301C + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MM	—	EOMIE	—	DTFLOWLVL		—		DTGTINT				—			
TYPE	R/W	R	R/W	R	R/W		R		R/W				R			
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**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxDATR contains the transaction attributes to be used for the message operation.

Table 16-108. OM*x*DATR Field Descriptions

Bits	Reset	Description	
MM 31	0	**Multicast Mode** When set, sends the message operation to all targets indicated by the multicast group and list. Messages are limited to one segment and 256 bytes or less.	0 Normal operation. 1 Multicast mode.
— 30	0	Reserved. Write to zero for future compatibility.	
EOMIE 29	0	**End-of-Message Interrupt Enable** When set, generates an interrupt when the current message operation finishes. For proper operation, this field should be modified only when the outbound message controller is not enabled.	
— 28	0	Reserved. Write to zero for future compatibility.	
DTFLOWLVL 27–26	0	**Transaction Flow Level** For proper operation, this field should be modified only when the outbound message controller is not enabled	00 Lowest priority transaction request flow. 01 Next highest priority transaction request flow. 10 Highest priority transaction request flow. 11 Reserved.
— 25–24	0	Reserved. Write to zero for future compatibility.	
DTGTINT 23–20	0	**Target Interface** Selects the target interface.	0000 SRIO Port 0 0001 SRIO Port 1 All other values reserved.
— 19–0	0	Reserved. Write to zero for future compatibility.	

16.6.65 Outbound Message x Double-Word Count Register (DMxDCR)

OM[0–1]DCR Outbound Message 0–1 Double-Word Count Registers — Offset 0x13020 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—			DCR										—		
TYPE	R			R/W										R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxDCR contains the number of double-words for the message operation. The maximum message operation size is 4 KB and the minimum is 8 bytes.

Table 16-109. OMxDCR Field Descriptions

Bits	Reset	Description	Settings
— 31–13	0	Reserved. Write to zero for future compatibility.	
DCR 12–3	0	**Transfer Count Register** Contains the number of bytes for the message operation.For proper operation, this field should be modified only when the outbound message controller is not enabled. The values with an asterisk (*) apply only in Multi-Segment mode. **Note:** The value in this register represents the number of double words to transfer and not the byte count; that is, the DCR value = the number of bytes/8.	00 0000 0000 Reserved. 00 0000 0001 8 bytes. 00 0000 0010 16 bytes. 00 0000 0100 32 bytes. 00 0000 1000 64 bytes. 00 0001 0000 128 bytes. 00 0010 0000 256 bytes. 00 0100 0000 512 bytes*. 00 1000 0000 1024 bytes*. 01 0000 0000 2048 bytes*. 10 0000 0000 4096 bytes*. All other values yield undefined behavior.
— 2–0	0	Reserved. Write to zero for future compatibility.	

16.6.66 Outbound Message x Descriptor Queue Enqueue Pointer Address Registers (OMxDQEPAR)

OM[0–1]DQEPAR Outbound Message x Descriptor Queue Enqueue Pointer Address Registers — Offset 0x13028 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DQEPA															
TYPE	R/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**
	DQEPA											—				
TYPE	R/W											R				
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxDQEPAR contains the address for the next descriptor in memory to be added to the queue. Software must initialize this register to match the outbound message descriptor queue dequeue pointer address. When a message is ready to be sent, the processor writes a descriptor to the next location in the queue (indicated by the address in OM*x*DQEPAR), and then either writes the OM*x*DQEPAR to point to the next descriptor location in memory or sets OM*x*MR[MUI]. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full. If the OM*x*MR[QFIE] bit is set, then the OM*x*SR[QFI] bit is set, and an interrupt is generated.
- If the enqueue and dequeue pointers no longer match after the enqueue pointer is incremented and the queue is full, then the queue overflows, and the message unit stops. If OM*x*MR[QOIE] is set, then the controller sets OM*x*SR[QOI] and generates an interrupt. OM*x*MR[MUS] must change from a 1 to a 0 to clear this error condition. If the enqueue pointer is written directly, the queue overflow condition is not detected.
- If the enqueue and dequeue pointers were the same before the register is incremented, the message unit controller will start, if enabled.

Note: When software initializes these registers, they must be aligned on a boundary equal to the number of queue entries × 32 bytes (the size of each queue descriptor). For example, if there are eight entries in the queue, the register must be 256-byte aligned. The number of queue entries is set in OMxMR[CIRQ_SIZ].

Table 16-110. OM*x*DQEPAR Field Descriptions

Bits	Reset	Description
DQEPA 31–5	0	**Descriptor Enqueue Pointer Address** Contains the address of the next free descriptor location. The descriptor must be aligned to a 32-byte boundary and a descriptor queue boundary.
— 4–0	0	Reserved. Write to zero for future compatibility.

16.6.67 Outbound Message x Retry Error Threshold Configuration Register (OMxRETCR)

OM[0–1]RETCR Outbound Message 0–1 Retry Error Threshold Configuration Registers — Offset 0x1302C + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—								RET							
TYPE	R								R/W							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxRETCR controls the number of times the message unit can attempt to transfer a message segment to a specific destination before reporting an error. A message segment is retransmitted if a RETRY response is received from the target.

Table 16-111. OM*x*RETCR Field Descriptions

Bits	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0	**Retry Error Threshold** The number of times the message unit can attempt to transmit a message segment to a specific target. For proper operation, this field should be modified only when the outbound message controller is not enabled	0x00 Disabled. 0x01 Message segment transmitted only 1 time. 0x02 Message segment transmitted up to 2 times. ... 0xFF Message segment transmitted up to 255 times.

16.6.68 Outbound Message x Multicast Group Registers (OMxMGR)

OM[0–1]MGR Outbound Message 0–1 Multicast Group Registers — Offset 0x13030 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—					EMG								MG		
TYPE	R					R/W										
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

OMxMGR contains a multicast group (MG) value and extended multicast group number (EMG) which, in combination with the multicast list register and the multicast enable (OM*x*MR[MM] described in **Table 16-103, *OMxMR Field Descriptions,*** on page 16-195), indicates which device IDs are targets of a multicast operation. The multicast group represents the most significant three bits (bits[7–5]) of the RapidIO device IDs that are destinations of the message operation. The multicast list indicates a list of targets within that group. Each individual bit, when encoded, determines the least significant five bits of the RapidIO device IDs (bits[4–0]) that are targets of the message operation. Therefore, multicast group 0 (MG = 0) contains target device IDs (0, 1, ..., 31), multicast group 1 (MG = 1) contains target device IDs (32, 33, ... 63), and so on.

In large transport mode, the extended multicast group represents the eight most significant bits (bits[15–8]), the multicast group represents the next three bits (bits[7–5]) and the multicast list indicates a list of targets within that group.

If multicast is enabled, this information in the multicast group and mask register is used to determine the target of the message operation instead of the DTGTROUTE and EDTGTROUTE fields in the Outbound Message Destination Port Register (see **Table 16-107, *OMxDPR Field Descriptions,*** on page 16-200).

Table 16-112. OM*x*MGR Field Descriptions

Bits	Name	Description
— 31–11	0	Reserved. Write to zero for future compatibility.

Table 16-112. OM*x*MGR Field Descriptions (Continued)

Bits	Name	Description
EMG 10–3	0	**Extended Multi-Cast Group** The most significant eight bits of the target device IDs for the multicast operation in Large Transport mode. For proper operation, this field should be modified only when the outbound message controller is not enabled
MG 2–0	0	**Multi-Cast Group** The most significant three bits of the target device IDs for the multicast operation. In Large Transport mode, these are the most significant bits of the least significant byte of the Target Device ID. For proper operation, this field should be modified only when the outbound message controller is not enabled

16.6.69 Outbound Message x Multicast List Registers (OMxMLR)

OM[0–1]MLR Outbound Message 0–1 Multicast List Registers Offset 0x13034 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ML															
TYPE	R/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**
	ML															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See **Section 16.6.68**, *Outbound Message x Multicast Group Registers (OMxMGR)*, on page 16-205 for more information.

Table 16-113. OM*x*MLR Field Descriptions

Bits	Reset	Description
ML 31–0	0	**Multicast List** The group target list for the message operation. Depending upon the value of the multicast group value, bit 31 corresponds to device ID 0, 32, 64, 96, and so on. Bit 30 corresponds to device ID 1, 33, 65, 97, and so on. If none of the bits are set, bit 31 is assumed to be set. For proper operation, this field should be modified only when the outbound message controller is not enabled.

16.6.70 Inbound Message x Mode Registers (IMxMR)

IM[0–1]MR Inbound Message 0–1 Mode Registers Offset 0x13060 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MIQ_THRESH				—								FRM_SIZ			
TYPE	R/W				R								R/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**
	CIRQ_SIZ				—			QFIE	—	MIQIE	EIE	—			MI	ME
TYPE	R/W				R			R/W	R	R/W		R			R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMxMR allows software to enable the mailbox controller and to control various characteristics of the message operation.

Table 16-114. IM*x*MR Field Descriptions

Bits	Reset	Description	Settings
MIQ_THRESH 31–28	0	**Message in Queue Threshold** Determines the number of message frames to be accumulated in the frame queue before Message In Queue is signaled. Results are undefined if the message in queue threshold is greater than or equal to the message queue size (IM*x*MR[CIRQ_SIZ]). For proper operation, this field should be modified only when the inbound message controller is not enabled.	0000 1. 0001 2. 0010 4. 0011 8. 0100 16. 0101 32. 0110 64. 0111 128. 1000 256. 1001 512. 1010 1024. 1011–1111 Reserved.
— 27–20	0	Reserved. Write to zero for future compatibility.	
FRM_SIZ 19–16	0	**Message Frame Size** Determines the maximum message size the mailbox can accept without error. Combined with IMxMR[CIRQ_SIZ], this parameter determines the maximum contiguous memory space allocated to the mailbox. For proper operation, this field should be modified only when the inbound message controller is not enabled.	0000–0001 Reserved. 0010 8 bytes. 0011 16 bytes. 0100 32 bytes. 0101 64 bytes. 0110 128 bytes. 0111 256 bytes. 1000 512 bytes. 1001 1024 bytes. 1010 2048 bytes. 1011 4096 bytes. 1100–1111 Reserved.

Table 16-114. IM*x*MR Field Descriptions (Continued)

Bits	Reset	Description	Settings
CIRQ_SIZ 15–12	0	**Circular Frame Queue Size** Determines the number of message frames that can be placed on the circular queue without overflow. Combined with IMxMR[FRM_SIZ], this parameter determines the maximum contiguous memory space allocated to the mailbox. This field should be modified only when the inbound message controller is not enabled.	0000 2. 0001 4. 0010 8. 0011 16. 0100 32. 0101 64. 0110 128. 0111 256. 1000 512. 1001 1024. 1010 2048. 1011–1111 Reserved.
— 11–9	0	Reserved. Write to zero for future compatibility.	
QFIE 8	0	**Queue Full Interrupt Enable** When set, the controller generates an interrupt when the queue is full (that is, the enqueue and dequeue pointers are equal after the mailbox controller increments the dequeue pointer). No QFI interrupt is generated if this if this bit is cleared. If this bit is set and IM*x*SR[QF] = 1, IM*x*SR[QFI] is set.	0 No interrupt is generated. 1 Interrupt generaated on queue full.
— 7	0	Reserved. Write to zero for future compatibility.	
MIQIE 6	0	**Message in Queue Interrupt Enable** When set, the controller enerates an interrupt when the queue has accumulated the number of messages specified by the IM*x*MR[MIQ_THRESH]. No MIQ interrupt is generated if this bit is cleared. If this bit is set and IM*x*SR[MIQ] = 1, IM*x*SR[MIQI] is set. If this bit is set and IM*x*MR[MI] is also set simultaneously, IM*x*SR[MIQI] reflects the value of MIQ after the increment.	0 No interrupt is generated. 1 Interrupt generaated on message in queue event.
EIE 5	0	**Error Interrupt Enable** When set, the controller generates a port-write/error interrupt when a transfer error (IM*x*SR[TE]) or a message request time-out (IM*x*SR[MRT]) event occurs. No port-write/error interrupt is generated if this bit is cleared.	0 No interrupt is generated. 1 Interrupt generaated on error.
— 4–2	0	Reserved. Write to zero for future compatibility.	
MI 1	0	**Mailbox Increment** Software sets this bit after processing an inbound message. Hardware increments the IM*x*FQDPAR and clears this bit. MI always reads as 0.	
ME 0	0	**Mailbox Enable** Set when the mailbox is initialized and can service incoming message operations. If this bit is cleared after the first segment of a multi-segment message arrives, a message request time-out results (IM*x*SR[MRT]). The busy bit (IM*x*SR[MB]) clears if the port response timer value (PRTOCCSR[TV]) is not set to the disabled value. If it is set to the disabled value, the busy bit does not clear.	

16.6.71 Inbound Message x Status Registers (IMxSR)

IM[0–1]SR Inbound Message 0–1 Status Registers Offset 0x13064 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—											QF	—			MIQ
TYPE	R															
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
	—					MRT	—		TE	—		QFI	—	MB	QE	MIQI
TYPE	R					W1C	R		W1C	R		W1C	R			W1C
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

IMxSR reports various mailbox conditions during and after a message operation. Writing a value of 1 to the corresponding set bit clears the bit.

Table 16-115. IM*x*SR Field Descriptions

Bits	Reset	Description
— 31–21	0	Reserved. Write to zero for future compatibility.
QF 20	0	**Queue Full** If the queue becomes full, this bit is set. QF is cleared when the queue is not full and if the message controller is disabled. Read only.
— 19–17	0	Reserved. Write to zero for future compatibility.
MIQ 16	0	**Message-In-Queue** If the queue has accumulated the number of messages specified by the IM*x*MR[MIQTH], this bit is set. MIQ is cleared when the number of message in the queue is less than the number specified by IMxMR[MIQ_THRESH] and if the message controller is disabled. Read only.
— 15–11	0	Reserved. Write to zero for future compatibility.
MRT 10	0	**Message Request Time-Out** Set when the message unit has not received another message segment for a multi-segment message and a time-out occurs. This bit is cleared by writing a 1 to it. For proper operation, this bit should be modified only when the inbound message controller is not enabled.
— 9–8	0	Reserved. Write to zero for future compatibility.
TE 7	0	**Transaction Error** Set when an internal error condition occurs during the message operation. TE is cleared by writing a 1 to it. For proper operation, this bit should be modified only when the inbound message controller is not enabled.
— 6–5	0	Reserved. Write to zero for future compatibility.

Table 16-115. IM*x*SR Field Descriptions (Continued)

Bits	Reset	Description
QFI 4	0	**Queue Full Interrupt** If the queue is full and the IMxMR[QFIE] bit is set, QFI is set and an interrupt is generated. QFI s cleared by writing a 1 to it.
— 3	0	Reserved. Write to zero for future compatibility.
MB 2	0	**Mailbox Busy** Indicates that a message operation is in progress. MB is cleared when an error occurs or the message operation finishes. Read only.
QE 1	1	**Queue Empty** If the queue is empty, this bit is set. QE is also set if the message controller is disabled. Read only.
MIQI 0	0	**Message-In-Queue Interrupt** If the queue has accumulated the number of messages specified by the IM*x*MR[MIQ_THRESH] and the IMxMR[MIQIE] bit is set, this bit is set and an interrupt is generated. This bit is cleared by writing a 1 to it.

16.6.72 Inbound Message x Frame Queue Dequeue Pointer Address Registers (IMxFQDPAR)

IM[0–1]FQDPAR Inbound Message 0–1 Frame Queue Dequeue Pointer Address Registers — Offset 0x1306C + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FQDPA															
TYPE	R/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**
	FQDPA													—		
TYPE	R/W													R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMxFQDPAR contains the address for the first message in memory to be processed. Software must initialize this register to the first frame location in memory. When a message is processed, the processor sets IM*x*MR[MI]. The mailbox hardware then increments IM*x*FQDPAR to point to the next frame in memory and clears the IM*x*MR[MI] bit. If the inbound message frame queue enqueue pointer and the inbound message frame queue dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next message frame from memory for processing. If the enqueue and dequeue pointers are equal after the processor increments them, the queue is empty and all outstanding messages are processed.

When software initializes thesee registers, they must be aligned on a boundary equal to the number of queue entries x frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the register must be 1024-byte aligned. The number of queue entries is set in IMxMR[CIRQ_SIZ] and the frame size is set in IMxMR[FRM_SIZ].

Table 16-116. IM*x*FQDPAR Field Descriptions

Bits	Reset	Description
FQDPA 31–3	0	**Frame Dequeue Pointer Address** Contains the address of the first message in memory to process.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.73 Inbound Message x Frame Queue Enqueue Pointer Address Registers (IMxFQEPAR)

IM[0–1]FQEPAR Inbound Message x Frame Queue Enqueue Pointer Address Registers — Offset 0x13074 + x*0x100

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FQEPA															
TYPE	R/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**
	FQEPA													—		
TYPE	R/W													R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IMxFQEPAR contains the address for the next message frame in memory to be added to the queue. Software must initialize this register to match the frame queue dequeue pointer address. When the mailbox controller receives a message, it writes the message data to the next location in the queue (indicated by the address in IM*x*FQEPAR) and then increments IM*x*FQEPAR to point to the next frame location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is full and the mailbox controller does not accept any more incoming messages, returning RETRY responses to the sending devices until the queue is no longer full. If the IM*x*MR[QFIE] bit is set, the IM*x*MR[QFI] bit is set and an interrupt is generated.
- If the enqueue and dequeue pointers are the same before the register is incrememted, the queue has changed from empty to not empty. If IM*x*MR[MIQIE] is set, IM*x*SR[MIQI] is set, and an interrupt is generated.

When software initializes these registers, they must be aligned on a boundary equal to the number of queue entries x frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the register must be 1024-byte aligned. The number of queues is set in IMxMR[CIRQ_SIZ] and the frame size is set in IMxMR[FRM_SIZ].

Table 16-117. IM*x*FQEPAR Field Descriptions

Bits	Reset	Description
FQEPA 31–3	0	**Frame Queue Enqueue Pointer Address** Contains the address of the next message frame to be added to the queue. For proper operation, this field should be modified only when the inbound message controller is not enabled.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.74 Inbound Message x Maximum Interrupt Report Interval Registers (IMxMIRIR)

IM[0–1]MIRIR Inbound Message 0–1 Maximum Interrupt Report Interval Registers — Offsert 0x13078 + x*0x100

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

IMxMIRIR contains a time-out timer value to define the maximum amount of time that the mailbox controller is to wait after transitioning from not empty before signaling an interrupt to the local processor (if enabled) if the IM*x*MR[MIQ_THRESH] limit is not reached. The reset value is the maximum time-out interval, and represents between 3 and 5 seconds.

Table 16-118. IM*x*MIRIR Field Descriptions

Bits	Reset	Description
MIRI 31–8	0xFFFFFF	**Maximum Interrupt Report Interval** Maximum interval between receiving the first message and setting IMxSR[IMIQ]. A value of 0 disables the time-out timer. For proper operation, this field should be modified only when the inbound message controller is not enabled.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.75 Outbound Doorbell Mode Register (ODMR)

ODMR Outbound Doorbell Mode Register Offset 0x13400

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—															DUS
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODMR allows software to start a doorbell operation and to control the characteristics of various doorbell operations.

Table 16-119. ODMR Field Descriptions

Bits	Reset	Description
— 31–1	0	Reserved. Write to zero for future compatibility.
DUS 0	0	**Doorbell Unit Start** A 0-to-1 transition when the doorbell unit is not busy (ODSR[DUB] bit has a value of 0) starts the doorbell unit. A 1-to-0 transition has no effect.

16.6.76 Outbound Doorbell Status Register (ODSR)

ODSR Outbound Doorbell Status Register Offset 0x13404

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—			MER	RETE	PRT	—							DUB	EODI	—
TYPE	R			W1C			R							R	W1C	R
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODSR reports various doorbell conditions during and after a doorbell operation. Writing a 1 to the corresponding set bit clears the bit.

Table 16-120. ODSR Field Descriptions

Bits	Reset	Description
— 31–13	0	Reserved. Write to zero for future compatibility.
MER 12	0	**Message Error Response** Set when an error response is received from the doorbell target. The error response reserved field indicates the value of the error response status bits when an error response is received. MER is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
RETE 11	0	**Retry Error Threshold Exceeded** Set when the doorbell unit cannot complete a doorbell operation because the retry error threshold value has been exceeded due to a RapidIO retry response. RETE is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
PRT 10	0	**Packet Response Time-Out** Set when the doorbell unit cannot complete a doorbell operation and a packet response time-out occurs. PRT is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
— 9–3	0	Reserved. Write to zero for future compatibility.
DUB 2	0	**Doorbell Unit Busy** Indicates that a doorbell operation is in progress. DUB is cleared when an error occurs or the doorbell operation is finishes. Read only.
EODI 1	0	**End-of-Doorbell Interrupt** When a doorbell operation finishes and the ODDATR[EODIE] bit is set, this bit is set and an interrupt is generated. EODI is cleared by writing a 1 to it. For proper operation, this bit should be cleared only when a doorbell operation is not in progress.
— 0	0	Reserved. Write to zero for future compatibility.

16.6.77 Outbound Doorbell Destination Port Register (ODDPR)

ODDPR Outbound Doorbell Destination Port Register Offset 0x13418

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EDTGROUTE								DTGTROUTE							
TYPE	R/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**
	—															
TYPE	R															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODDPR indicates the RapidIO destination device ID to which the doorbell unit controller is to send data. Software must ensure that this is a valid port in the receiving device.

Table 16-121. ODDPR Field Descriptions

Bits	Reset	Description
EDTROUTE 31–24	0	**Extended Destination Target Route** Most significant byte of a 16-bit target route (device ID of the target) in Large Transport mode. In Small Transport mode, this bit is reserved. For proper operation, this field should be modified only when a doorbell operation is not in progress.
DTROUTE 23–16	0	**Destination Target Route** Contains the target route field of the transaction (device ID of the target). For proper operation, this field should be modified only when a doorbell operation is not in progress.
— 15–0	0	Reserved. Write to zero for future compatibility.

16.6.78 Outbound Doorbell Destination Attributes Register (ODDATR)

ODDATR Outbound Doorbell Destination Attributes Register Offset 0x1341C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		EODIE	—	DTFLOWLVL		—		TGINT				—			
TYPE	R/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**
	INFO_MSB								INFO_LSB							
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODDATR contains the transaction attributes to be used for the doorbell operation.

Table 16-122. ODDATR Field Descriptions

Bits	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility. Bit 30 is hardwired to 0.	
EODIE 29	0	**End-of-Doorbell Interrupt Enable** Generates an interrupt when the current doorbell operation finishes. This field should be modified only when a doorbell operation is not in progress.	
— 28	0	Reserved. Write to zero for future compatibility.	
DTFLOWLVL 27–26	0	**Transaction Flow Level** Specifies the transaction flow level. This field should be modified only when a doorbell operation is not in progress.	00 Lowest-priority transaction flow. 01 Next highest priority transaction flow. 10 Highest-priority transaction flow. 11 Reserved.
— 25–24	0	Reserved. Write to zero for future compatibility.	
TGINT 23–20	0	**Target Interface** Selects the target interface.	0000 SRIO Port 0 0001 SRIO Port 1 All other values reserved.
— 19–16	0	Reserved. Write to zero for future compatibility.	
INFO_MSB 15–8	0	**MSB of Doorbell INFO** Most significant byte of the doorbell INFO field. This field should be modified only when a doorbell operation is not in progress.	
INFO_LSB 7–0	0	**LSB of Doorbell INFO** Least significant byte of the doorbell INFO field. This field should be modified onlyy when a doorbell operation is not in progress.	

16.6.79 Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)

ODRETCR Outbound Doorbell Retry Error Threshold Configuration Register — Offset 0x1342C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—								RET							
TYPE	R								R/W							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ODRETCR controls the number of times the doorbell unit attempts to complete a doorbell operation before reporting an error and moving on to the next task, if available. Failures to complete an operation are indicated when the doorbell unit receives a RapidIO logical layer RETRY response from the target. If the programmed count is exceeded, the ODSR[RETE] bit is set.

Table 16-123. ODRETCR Field Descriptions

Bits	Name	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RET 7–0	0	**Retry Error Threshold** Specifies the number of times the doorbell unit attempts to transmit a doorbell message. This field should be modified only when a doorbell operation is not in progress.	0x00 Disabled. 0x01 Doorbell transmitted only 1 time. 0x02 Doorbell transmitted up to 2 times ... 0xFF - Doorbell transmitted up to 255 times.

16.6.80 Inbound Doorbell Mode Registers (IDMR)

IDMR Inbound Doorbell Mode Register Offset 0x13460

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DIQ_THRESH				—											
TYPE	R/W				R											
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**
	CIRQ_SIZ				—			QFIE	—	DIQIE	EIE	—			DI	DE
TYPE	R/W				R			R/W	R	R/W		R			R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IDMR allows software to enable the inbound doorbell controller and to control the characteristics of various doorbell operations.

Table 16-124. IDMR Field Descriptions

Bits	Reset	Description	Settings
DIQ_THRESH 31–28	0	**Doorbell-in-Queue Threshold** Determines the number of doorbells to accumulate in the doorbell queue before the doorbell-in-queue bit is set (IDSR[DIQ]). Undefined operation results if the actual number of entries in the doorbell-in-queue threshold is set as greater than or equal to the actual size of the doorbell queue. For proper operation, this field should be modified only when the doorbell controller is not enabled.	0000 1. 0001 2. 0010 4. 0011 8. 0100 16. 0101 32. 0110 64. 0111 128. 1000 256. 1001 512. 1010 1024. 1011–1111 Reserved.
— 27–16	0	Reserved. Write to zero for future compatibility.	
CIRQ_SIZ 15–12	0	**Circular Doorbell Queue Size** Determines the number of doorbell entries that can be placed in the circular queue. CIRQ_SIZ × 8 bytes determine the maximum contiguous memory space allocated to the doorbell unit. For proper operation, this field should be modified only when the doorbell controller is not enabled.	0000 2. 0001 4. 0010 8. 0011 16. 0100 32. 0101 64. 0110 128. 0111 256. 1000 512 (4 KB page boundary). 1001 1024. 1010 2048. 1011–1111 Reserved.

Table 16-124. IDMR Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 11–9	0	Reserved. Write to zero for future compatibility.	
QFIE 8	0	**Queue Full Interrupt Enable** When set, enables the queue full interrupt. The queue is full when the enqueue and dequeue pointers are equal after the doorbell controller increments the dequeue pointer, the controller genrates the Queue Full interrupt. That is, if this bit is set and IDSR[QF] = 1, IDSR[QFI] is set. For proper operation, this field should only be modified when the inbound doorbell controller is not enabled.	0 No QF interrupt is generated. 1 Generates an interrupt when the queue is full .
— 7	0	Reserved. Write to zero for future compatibility.	
DIQIE 6	0	**Doorbell in Queue Interrupt Enable** When set, enables the doorbell in queue interrupt. The interrupt cannot be asserted if this bit is cleared. If this bit is set and IDSR[DIQ] = 1, IDSR[DIQI] is set. If this bit is set and IDMR[DI] is set simultaneously, IDSR[DIQI] reflects the value of DIQ after the increment. For proper operation, this field should only be modified when the inbound doorbell controller is not enabled.	0 No DIQ interrupt is generated. 1 Generates an interrupt when IDSR[DIQ] = 1.
EIE 5	0	**Error Interrupt Enable** When set, enables the port-write/error interrupt when a transfer error (IDSR[TE]) event occurs. No port-write/error interrupt is generated if this bit is cleared. For proper operation, this field should only be modified when the inbound doorbell controller is not enabled.	
— 4–2	0	Reserved Reserved. Write to zero for future compatibility.	
DI 1	0	**Doorbell Increment** Software sets this bit after an inbound doorbell is processed. Hardware then increments the IDQDPAR and clears this bit. DI always reads as 0.	
DE 0	0	**Doorbell Enable** Enabled/disables the inbound doorbell operations.	0 Inbound doorbell disabled. 1 The inbound doorbell is initialized and can service incoming doorbell operations.

16.6.81 Inbound Doorbell Status Register (IDSR)

IDSR Inbound Doorbell Status Register Offset 0x13464

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												QF	—			DIQ
TYPE	R															
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**
	—								TE	—		QFI	—	DB	QE	DIQI
TYPE	R								W1C	R		W1C	R			W1C
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

IDSR reports various doorbell conditions after a doorbell operation. Writing a 1 to the corresponding set bit clears the bit.

Table 16-125. IDSR Field Descriptions

Bits	Reset	Description
— 31–21	0	Reserved. Write to zero for future compatibility.
QF 20	0	**Queue Full** If the queue is full, this bit is set. This bit is cleared when the queue is not full or if the doorbell controller is disabled. Read only.
— 19–17	0	Reserved. Write to zero for future compatibility.
DIQ 16	0	**Doorbell-In-Queue** If the queue has accumulated the number of doorbells specified by DIQ_THRESH, then this bit is set. Also, if a valid entry pointed to by the dequeue address pointers has not been serviced within the configured maximum interval, this bit is set. This bit is cleared if the above conditions are not met or the doorbell controller is disabled. Read-only.
— 15–8	0	Reserved. Write to zero for future compatibility.
TE 7	0	**Transaction Error** Set when an internal error occurs during the doorbell operation. To reset TE, write a value of 1 to clear it. For proper operation, this bit should be modified only when the doorbell controller is not enabled..
— 6–5	0	Reserved. Write to zero for future compatibility.
QFI 4	0	**Queue Full Interrupt** If the queue becomes full and the QFIE bit in the Mode Register is set, this bit is set and an interrupt is generated. This bit is cleared by writing a 1 to it. For proper operation, this bit should be modified only when the doorbell controller is not enabled..
— 3	0	Reserved. Write to zero for future compatibility.
DB 2	0	**Doorbell Busy** Indicates that a doorbell has been received and the doorbell queue is being written. This bit is cleared when the write to memory finishes. Disabling the doorbell controller does not affect this bit. Read only.

Table 16-125. IDSR Field Descriptions (Continued)

Bits	Reset	Description
QE 1	1	**Queue Empty** If the queue is empty, then this bit is set. This bit will also be set if the doorbell controller is disabled. Read only.
DIQI 0	0	**Doorbell-In-Queue Interrupt** If DIQ is set and IDMR[DIQIE} is set, the controller sets this bit and generates an interrupt. This bit is cleared by writing a 1 to it.

16.6.82 Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR)

IDQDPAR Inbound Doorbell Queue Dequeue Pointer Address Registers Offset 0x1346C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DQDPA															
TYPE	R/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**
	DQDPA													—		
TYPE	R													R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IDQDPAR contains the double-word address for the first doorbell in memory to be processed. Software must initialize this register to the first doorbell entry location in memory. After a doorbell is processed, the processor sets IDMR[DI]. Then the doorbell queue dequeue pointer address register is incremented by hardware to point to the next doorbell entry in memory and IDMR[DI] is cleared.

When processing multiple doorbells, the processor can write this register directly instead of setting IDMR[DI] for each doorbell. If the enqueue pointer and the dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next doorbell from memory for processing. If the enqueue and dequeue pointers are equal after the processor increments the IDQDPAR, the queue is empty, and all outstanding doorbells have been processed.

Table 16-126. IDQDPAR Field Descriptions

Bits	Reset	Description
DQDPA 31–3	0	**Doorbell Dequeue Pointer Address** Contains the double-word address of the first doorbell in memory to process.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.83 Inbound Doorbell Queue Enqueue Pointer Address Registers (IDQEPAR)

IDQEPAR Inbound Doorbell Queue Enqueue Pointer Address Registers Offset 0x13474

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DQEPA															
TYPE	R/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**
	DQEPA													—		
TYPE	R													R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IDQEPAR contains the double-word address for the next doorbell entry in memory to be added to the queue. Software must initialize IDQEPAR to match the doorbell queue dequeue pointer address. When a doorbell packet is received by the doorbell controller, it writes the doorbell information to the next location in the queue (indicated by the address in IDQEPAR) and then increments IDQEPAR to point to the next doorbell location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, then the queue is now full and the doorbell controller will not accept any more incoming doorbell packets, returning RETRY responses to the sending devices until the queue is no longer full. If the IDMR[QFIE] bit is set, then the IDSR[QFI] is set and the interrupt is generated.
- If the enqueue and dequeue pointers were the same before receiving the doorbell, the queue has changed from empty to not empty. When the number of doorbells received matches the configured threshold, the IDSR[DIQ] bit is set. If the IDMR[DIQIE] bit is set, then the IDSR[DIQI] bit is also set and the inbound doorbell interrupt is generated.

Table 16-127. IDQEPAR Field Descriptions

Bits	Name	Description
DQEPA 31–3	0	**Doorbell Queue Enqueue Pointer Address** Contains the double-word address of the next doorbell location to be added to the queue. For proper operation, this field should be written only when the doorbell controller is disabled.
— 2–0	0	Reserved. Write to zero for future compatibility.

16.6.84 Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)

IDMIRIR Inbound Doorbell Maximum Interrupt Report Interval Register — Offset 0x13478

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

IDMIRIR contains a time-out timer value to define the maximum amount of time that a doorbell entry can be at the head of the doorbell queue before generating an interrupt if the IDMR[DIQ_THRESH] limit is not reached. The reset value is the maximum time-out interval and represents 3–6 seconds.

Table 16-128. IDMIRIR Field Descriptions

Bits	Reset	Description
MIRI 31–8	0xFFFFFF	**Maximum Interrupt Report Interval** Maximum interval between the time a doorbell message goes into the queue until an interrupt is generated. A value of 0 disables the timer. This field should be written only when the doorbell controller is disabled.
— 7–0	0	Reserved. Write to zero for future compatibility.

16.6.85 Inbound Port-Write Mode Register (IPWMR)

IPWMR Inbound Port-Write Mode Register Offset 0x134E0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
TYPE	R															
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**
	—							QFIE	—		EIE	—			CQ	PWE
TYPE	R							R/W	R		R/W	R			R/W	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPWMR allows software to enable the port-write controller and to control various characteristics of port-write operations.

Table 16-129. IPWMR Field Descriptions

Bits	Reset	Description
— 31–9	0	Reserved. Write to zero for future compatibility.
QFIE 8	0	**Queue Full Interrupt Enable** When set, enable a error/port-write interrupt when the queue is full; that is, the controller has written the port-write data payload into memory. No interrupt is generated if this if this bit is cleared. For proper operation, this field should be modified only when the port-write controller is not enabled.
— 7–6	0	Reserved. Write to zero for future compatibility.
EIE 5	0	**Error Interrupt Enable** When set, enables a port-write/error interrupt when a transfer error (IPW0SR[TE]) event occurs. No interrupt is generated if this bit is cleared. For proper operation, this field should be modified only when the port-write controller is not enabled.
— 4–2	0	Reserved. Write to zero for future compatibility.
CQ 1	0	**Clear Queue** Software sets this bit after processing an inbound port-write operation. Hardware clears the queue full bit (IPWSR[QF]), clears this bit, and allows another port-write to be received. This bit is always read as a 0.
PWE 0	0	**Port Write Enable** If this bit is set the port-write controller is initialized and can service an incoming operation.

16.6.86 Inbound Port-Write Status Register (IPWSR)

IPWSR Inbound Port-Write Status Register Offset 0x134E4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—											QF	—			
TYPE	R															
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**
	—								TE	—		QFI	PWD	PWB	—	
TYPE	R								W1C	R		W1C		R		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPWSR reports various port-write conditions.

Table 16-130. IPWSR Field Descriptions

Bits	Reset	Description	Settings
— 31–21	0	Reserved. Write to zero for future compatibility.	
QF 20	0	**Queue Full** Set when the queue becomes full. QF is cleared when the clear queue bit is set (IPWMR[CQ]) and the queue is not full. This bit is cleared when the queue is not full or if the port-write controller is disabled. Read only.	
— 19–16	0	Reserved. Write to zero for future compatibility.	
TE 15	0	**Transaction Error** Set when an internal error condition occurs during the port-write operation. Write a value of 1 to TE to clear it. This bit should be modified only when the port-write controller is not enabled.	
— 6–5	0	Reserved. Write to zero for future compatibility.	
QFI 4	0	**Queue Full Interrupt** If the queue is full and the IPWMR[QFIE] bit is set, this bit is set and an interrupt generated. This bit is cleared by writing a 1 to it.	
PWD 3	0	**Port-Write Discarded** Set when a port-write is discarded while the port-write controller is enabled but busy. This bit is cleared by writing a 1 to it.	
PWB 2	0	**Port-Write Busy** Indicates a port-write busy condition. Disabling the port-write controller does not affect this bit. Read only.	0 Port-write payload has been written to memory. 1 A port-write has been received and the port-write payload is being written to memory.
— 1–0	0	Reserved. Write to zero for future compatibility.	

16.6.87 Inbound Port-Write Queue Base Address Register (IPWQBAR)

IPWQBAR Inbound Port-Write Queue Base Address Register Offset 134EC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PWQBA															
TYPE	R/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**
	PWQBA										—					
TYPE	R/W										R					
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IPWQBAR contains the 64-byte cache line address for the port-write data payload. Software must initialize this register to the desired location in memory.

Table 16-131. IPWQBAR Field Descriptions

Bits	Reset	Description
PWQBA 31–6	0	**Port-Write Queue Base Address** Contains the address of the port-write data payload. This field should be written only when the port-write controller is disabled
— 5–0	0	Reserved. Write to zero for future compatibility.

PCI Express Controller 17

The PCI Express controller supports communication with PCI Express devices connected to the MSC8156E device. The PCI Express interface is designed to comply with the *PCI Express™ Base Specification, Revision 1.0a* (available from http://www.pcisig.org). This chapter describes the PCI Express controller and provides a basic description of the PCI Express protocol. Designers of systems incorporating PCI Express devices should refer to the specification for a thorough description of PCI Express structure and functionality.

17.1 Overview

The PCI Express controller connects the MSC8156E device through the High Speed Serial Interface (HSSI) to a 2.5-GHz serial interface configurable for up to a x4 interface. As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. When selected and enabled by the device configuration, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner after it completes its reset sequence. Once link autonegotiation is successful, the controller is available to transfer data. In x4 mode only, the controller permits lanes to be reversed between the transmitting and the receiving device (that is, transmitter lanes 0-1-2-3 are swapped to 3-2-1-0 of the receiver).

Note: See **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem* for details on signal multiplexing and data communications with the MSC8156E cores, memory, and internal peripherals.

Internally, the design contains queues to keep track of inbound and outbound transactions. Control logic handles buffer management, bus protocol, transaction spawning and tag generation. In addition, memory blocks store inbound and outbound data. **Figure 17-1** is a high-level block diagram of the PCI Express controller.

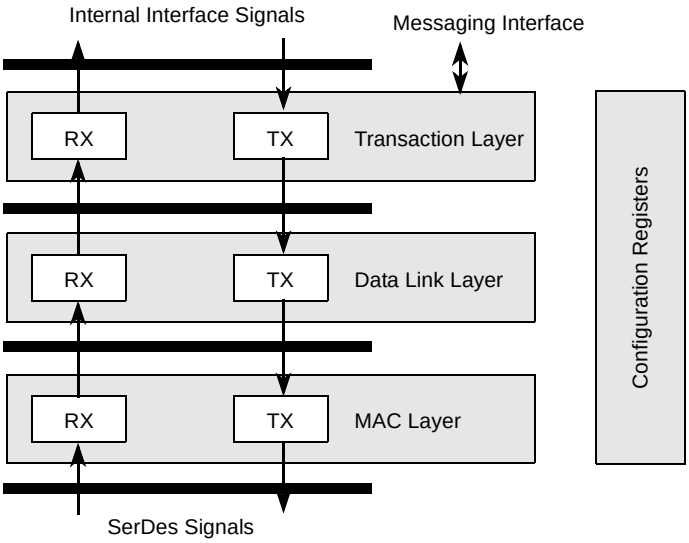


Figure 17-1. PCI Express Controller Block Diagram

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 core processor/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.

As an initiator, the PCI Express controller supports memory read and write operations with a maximum transaction size of 256 bytes. In RC mode, the controller also supports configuration and I/O transactions. 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.

17.1.1 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. 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, 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: 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_LTSSM_STAT) to check the status of link training before issuing external requests.

17.1.2 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 to send next to the internal interface.

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: The controller splits transactions at the crossing of every 256-byte-aligned boundary when sending data back to the PCI Express link.

17.2 Features

The PCI Express controller includes the following features:

- Designed to comply with the *PCI Express™ Base Specification, Revision 1.0a.*
- Supports root complex (RC) and endpoint (EP) configurations.
- 32- and 64-bit address support.
- x4, x2, or x1 link support.
- Supports accesses to all PCI Express memory and I/O address spaces (requestor only).
- Supports posting of processor-to-PCI Express and PCI Express-to-memory writes.
- Supports strong and relaxed transaction ordering rules.
- 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 and one configuration window.
- Supports four outbound translation windows and one default window.
- Supports eight non-posted and four posted PCI Express transactions.
- Supports up to six priority 0 internal platform reads and eight priority 0 to 2 internal platform writes. The maximum number of outstanding transactions at any given time is eight.
- Credit-based flow control management.
- Supports PCI Express messages and interrupts.
- Accepts up to 256-byte transactions.

17.3 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.

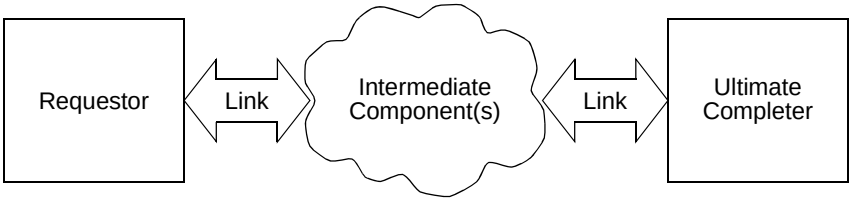


Figure 17-2. Requestor/Completer Relationship

Each PCI Express device is divided into two halves: transmit (TX) and receive (RX). Each half is further divided into three layers: transaction, data link, and physical, as shown in **Figure 17-3**.

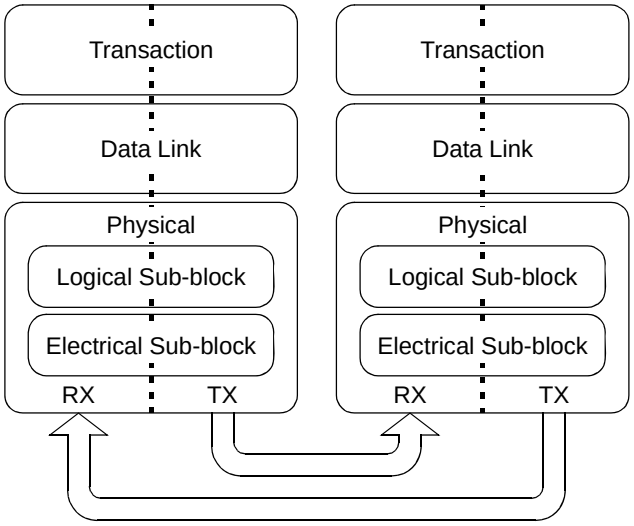


Figure 17-3. PCI Express High-Level Layering

Packets are formed in the transaction layer (TLPs) and data link layer (DLLPs). Each subsequent layer adds the necessary encodings and framing, illustrated in **Figure 17-4**. As packets are received, they are decoded and processed by the same layers, but in reverse order for processing by the layer or by the device application software.

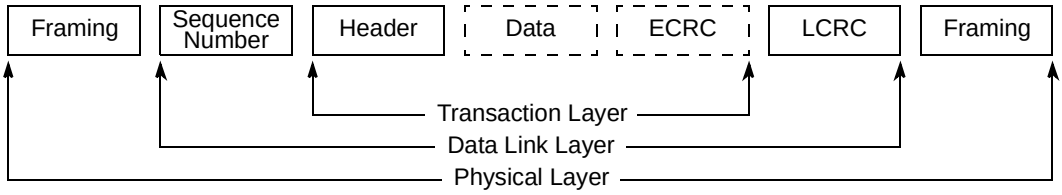


Figure 17-4. PCI Express Packet Flow

17.3.1 Modes of Operation

The PCI Express controller can function as either a root complex (RC) or an endpoint (EP) on the PCI Express link. The mode and port width are selected by reset configuration signals (see **Chapter 5**, *Reset* for details.

17.3.2 PCI Express Transactions

Table 17-1 lists the transactions that the PCI Express controller supports as an initiator and a target.

Table 17-1. 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
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
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

17.3.2.1 Byte Ordering

Whenever data must cross a bridge between two buses, 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 byte ordering policy.

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 17-5 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.

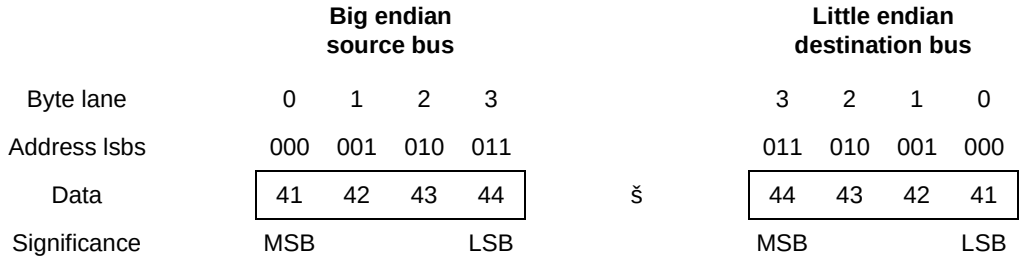


Figure 17-5. Address Invariant Byte Ordering—4 bytes Outbound

Note: Although the significance of the bytes within the scalar has changed, the address of the individual bytes that make up the scalar has 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 17-6 shows data flowing the other way, from a little endian source to a big endian destination.

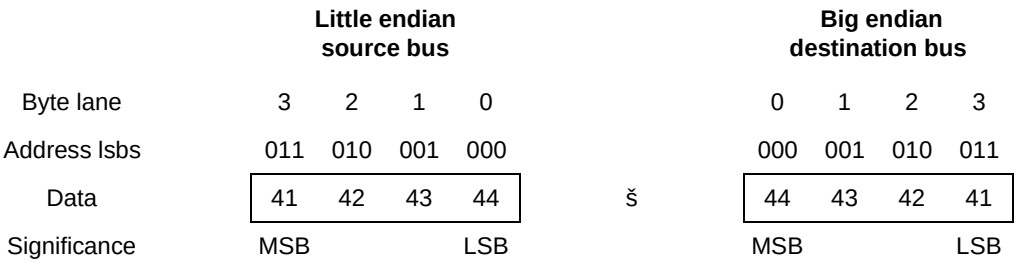


Figure 17-6. Address Invariant Byte Ordering—4 bytes Inbound

Figure 17-7 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

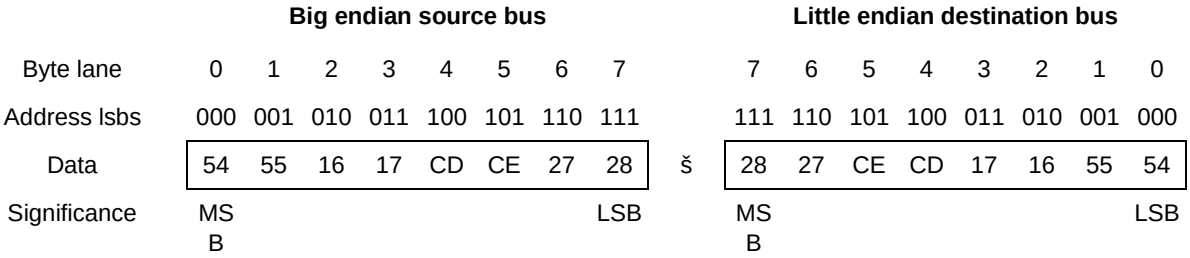


Figure 17-7. Address Invariant Byte Ordering—8 bytes Outbound

Figure 17-8 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

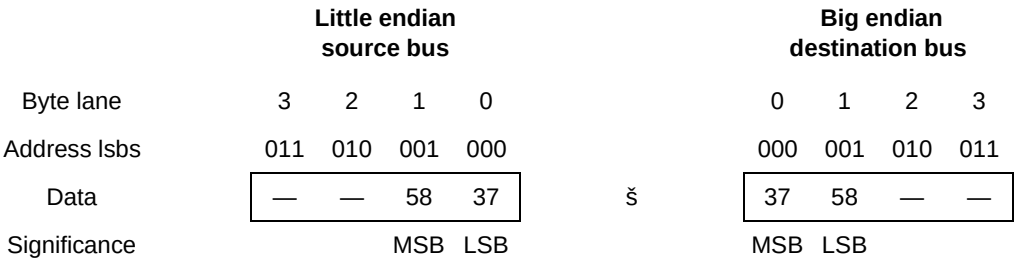


Figure 17-8. Address Invariant Byte Ordering—2 bytes Inbound

Note: In all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) are preserved.

17.3.2.2 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 17-9**. Therefore, software must access PEX_CONFIG_DATA with little-endian formatted data—either by using the **swapb** instruction or by manipulating the data before writing to and after reading from PEX_CONFIG_DATA.

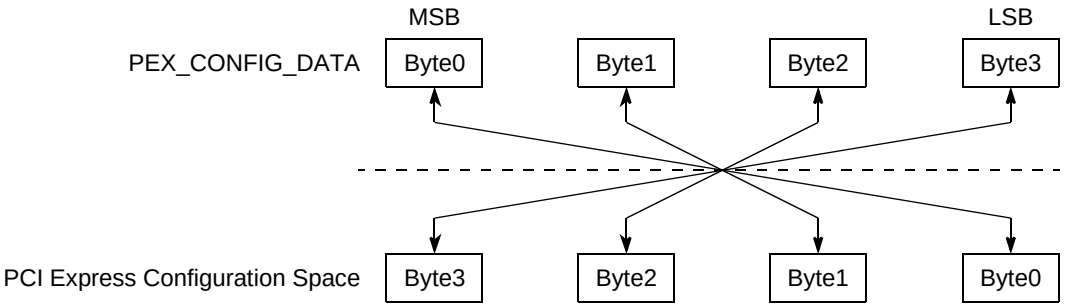


Figure 17-9. PEX_CONFIG_DATA Byte Ordering

17.3.2.3 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 will bypass all other transactions except another posted request.
- A completion can and will only bypass non-posted requests. It can and will bypass 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.

17.3.2.4 Memory Space Addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space. **Table 17-2** describes the Fmt and Type field contents in the PCI Express TLP header for memory transactions.

Table 17-2. TLP Header Type and Format Field Definitions for Memory Transactions

Request	Fmt[1–0]	Type[4–0]
Mem read (32-bit)	00	00000
Mem read (64-bit)	01	00000
Mem write (32-bit)	10	00000
Mem write (64-bit)	11	00000

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.

17.3.2.5 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 by programming one of the outbound translation window attributes to send I/O transactions. All I/O transactions only access 32-bit address I/O space. **Table 17-3** describes the Fmt and Type field contents in the PCI Express TLP header for I/O transactions.

Table 17-3. TLP Header Type and Format Field Definitions for I/O Transactions

Request	Fmt[1–0]	Type[4–0]
I/O read (32-bit)	00	00010
I/O write (32-bit)	10	00010

17.3.2.6 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 17.4.1.6, *PCI Express Configuration Space Access, on page 17-65***, for more information. A configuration transaction can hit into the controller internal configuration space, it can be sent out on the PCI Express link, or it can be internally terminated. **Table 17-4** describes the Fmt and Type field contents in the PCI Express TLP header for configuration transactions.

Table 17-4. TLP Header Type and Format Field Definitions for Configuration Transactions

Request	Fmt[1–0]	Type[4–0]
Config Type 0 read	00	00100
Config Type 0 write	10	00100
Config Type 1 read	00	00101
Config Type 1 write	10	00101

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.

17.3.2.7 Serialization of Configuration and I/O Writes

Configuration and I/O writes originating from the PCI Express outbound ATMUs are serialized by the controller. The logic after issuing a configuration write or IO write will 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 will re-issue the configuration write transaction. It will keep retrying the request until either a status other than CRS is returned or the transaction times out. Note that configuration writes originating from the PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA) are not serialized.

Note: 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.

17.3.3 Messages

Software message generation is supported in both RC and EP modes.

17.3.3.1 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. **Table 17-5** describes the message data format.

Table 17-5. Internal Platform (OCN) Message Data Format

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.

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 17.4.1.2.4**, *PCI Express Power Management Command Register (PEX_PMCR)*, on page 17-36 for more information.

Table 17-6 provides a complete list of supported outbound messages depending on whether RC or EP is configured.

Table 17-6. 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	000	No	N/A	Not supported
Set_Slot_Power_Limit	0101 0000	100	Yes	N/A	Set Slot Power Limit in Upstream Port
Vendor_Defined Type 0	0111 1110		No	No	Not supported
Vendor_Defined Type 1	0111 1111		No	No	Not supported
Attention_Indicator_On	0100 0001	100	Yes	N/A	Hot-plug message
Attention_Indicator_Blink	0100 0011	100	Yes	N/A	Hot-plug message
Attention_Indicator_Off	0100 0000	100	Yes	N/A	Hot-plug message
Power_Indicator_On	0100 0101	100	Yes	N/A	Hot-plug message
Power_Indicator_Blink	0100 0111	100	Yes	N/A	Hot-plug message
Power_Indicator_Off	0100 0100	100	Yes	N/A	Hot-plug message
Attention_Button_Pressed	0100 1000	100	Yes	N/A	Hot-plug message

17.3.3.2 Inbound Messages

Table 17-7 provides a complete list of supported inbound messages in RC mode.

Table 17-7. PCI Express RC Inbound Message Handling

Name	Code[7–0]	Routing[2–0]	Action
Assert_INTA	0010 0000	100	Send to interrupt controller
Assert_INTB	0010 0001	100	Send to interrupt controller
Assert_INTC	0010 0010	100	Send to interrupt controller
Assert_INTD	0010 0011	100	Send to interrupt controller
Deassert_INTA	0010 0100	100	Send to interrupt controller
Deassert_INTB	0010 0101	100	Send to interrupt controller
Deassert_INTC	0010 0110	100	Send to interrupt controller
Deassert_INTD	0010 0111	100	Send to interrupt controller
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	Generate interrupt to interrupt controller 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 interrupt controller if enabled
ERR_COR	0011 0000	000	Generate interrupt to interrupt controller if enabled
ERR_NONFATAL	0011 0001	000	Generate interrupt to interrupt controller if enabled
ERR_FATAL	0011 0011	000	Generate interrupt to interrupt controller 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
Attention_Indicator_On	0100 0001	100	No action taken
Attention_Indicator_Blink	0100 0011	100	No action taken
Attention_Indicator_Off	0100 0000	100	No action taken
Power_Indicator_On	0100 0101	100	No action taken
Power_Indicator_Blink	0100 0111	100	No action taken
Power_Indicator_Off	0100 0100	100	No action taken
Attention_Button_Pressed	0100 1000	100	Set PEX_PME_MES_DR[ABP] bit and send interrupt if enabled.

Table 17-8 provides a complete list of supported inbound messages in EP mode.

Table 17-8. 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

Table 17-8. PCI Express EP Inbound Message Handling (Continued)

Name	Code[7–0]	Routing[2–0]	Action
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
Attention_Indicator_On	0100 0001	100	Set PEX_PME_MES_DR[AION] bit. Send interrupt if enabled.
Attention_Indicator_Blink	0100 0011	100	Set PEX_PME_MES_DR[AIB] bit. Send interrupt if enabled.
Attention_Indicator_Off	0100 0000	100	Set PEX_PME_MES_DR[AIOF] bit. Send interrupt if enabled.
Power_Indicator_On	0100 0101	100	Set PEX_PME_MES_DR[PION] bit. Send interrupt if enabled.
Power_Indicator_Blink	0100 0111	100	Set PEX_PME_MES_DR[PIB] bit. Send interrupt if enabled.
Power_Indicator_Off	0100 0100	100	Set PEX_PME_MES_DR[PIOF] bit. Send interrupt if enabled.
Attention_Button_Pressed	0100 1000	100	No action taken

17.3.4 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 17-10** uncorrectable errors can further be classified as fatal or non-fatal.

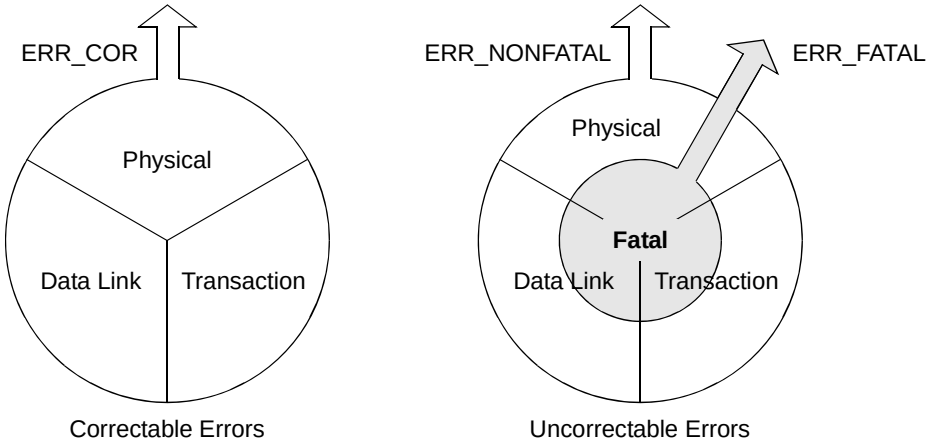


Figure 17-10. PCI Express Error Classification

17.3.4.1 PCI Express Error Logging and Signaling

Figure 17-11 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.

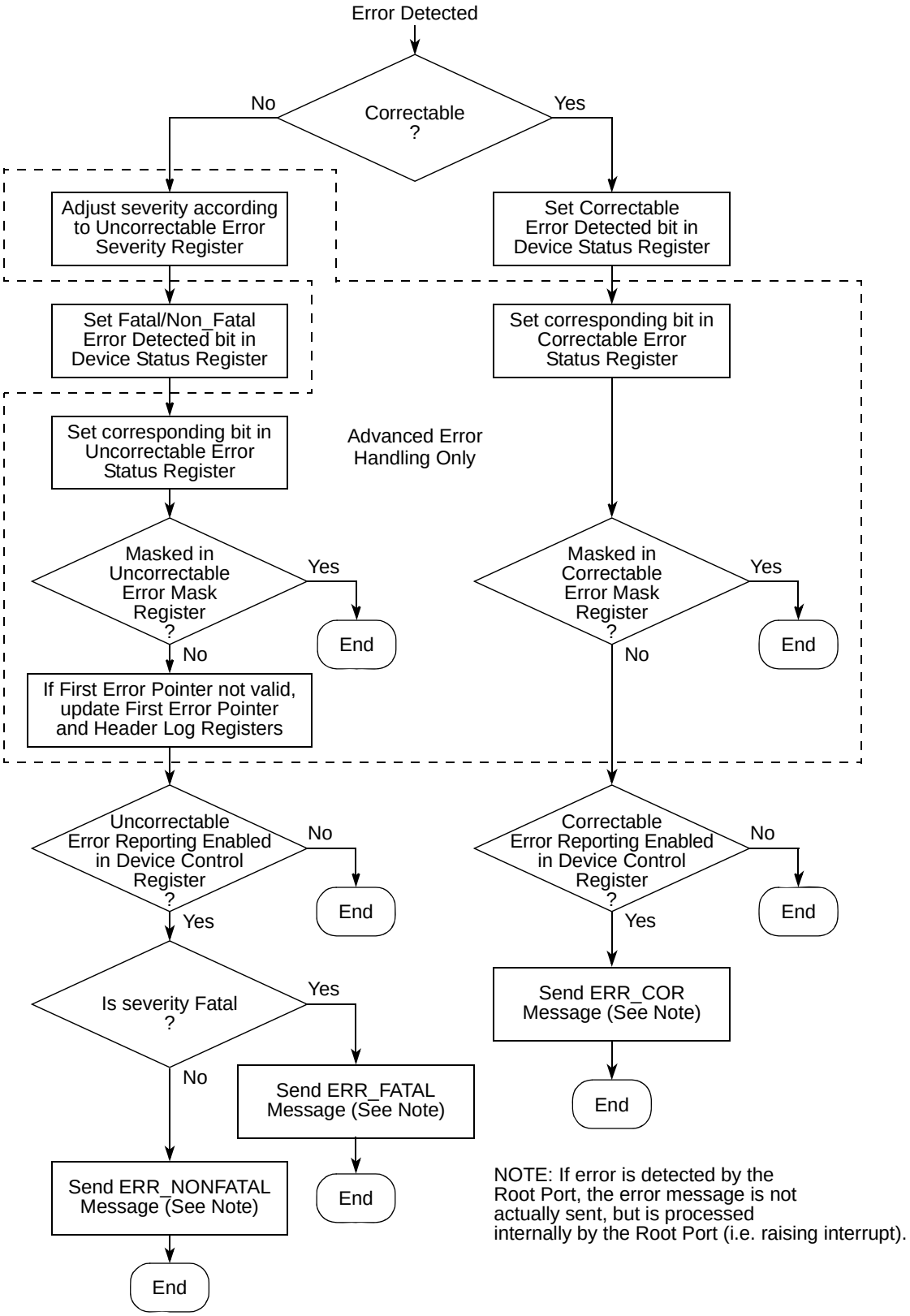


Figure 17-11. PCI Express Device Error Signaling Flowchart

17.3.4.2 PCI Express Controller Internal Interrupt Sources

Table 17-9 describes the sources of the PCI Express controller internal interrupt to the EPIC and the preconditions for signaling the interrupt.

Table 17-9. 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	PCI Express Root Control Register [3] (PME interrupt enable) is set
PCI Express Root Error Status Register[6] (fatal error messages received) is set	PCI Express Root Error Command Register [2] (fatal error reporting enable) is set or PCI Express Root Control Register [2] (system error on fatal error enable) is set
PCI Express Root Error Status Register [5] (non-fatal error messages received) is set	PCI Express Root Error Command Register [1] (non-fatal error reporting enable) is set or PCI Express Root Control Register [1] (system error on non-fatal error enable) is set
PCI Express Root Error Status Register[0] (correctable error messages received) is set	PCI Express Root Error Command Register[0] (correctable error reporting enable) is set or PCI Express Root Control Register[0] (system error on correctable error enable) is set.
Any correctable error status bit in PCI Express Correctable Error Status Register is set	The corresponding error mask bit in PCI Express Correctable Error Mask Register is clear and PCI Express Root Error Command Register[0] (correctable error reporting enable) is set
Any fatal uncorrectable error status bit in PCI Express Uncorrectable Error Status Register is set. (The corresponding error is classified as fatal based on the severity setting in PCI Express Uncorrectable Error Severity Register.)	The corresponding error mask bit in PCI Express Uncorrectable Error Mask Register is clear and either PCI Express Device Control Register[2] (fatal error reporting) is set or PCI Express Command Register[8] (SERR) is set.
Any non-fatal uncorrectable error status bit in PCI Express Uncorrectable Error Status Register is set. (The corresponding error is classified as non-fatal based on the severity setting in PCI Express Uncorrectable Error Severity Register.)	The corresponding error mask bit in PCI Express Uncorrectable Error Mask Register is clear and either PCI Express Device Control Register[1] (non-fatal error reporting) is set or PCI Express Command Register[8] (SERR) is set.
PCI Express Secondary Status Register[8] (master data parity error) is set.	PCI Express Secondary Status Interrupt Mask Register[0] (mask master data parity error) is cleared and PCI Express Command Register[6] (parity error response) is set.
PCI Express Secondary Status Register[11] (signaled target abort) is set	PCI Express Secondary Status Interrupt Mask Register[1] (mask signaled target abort) is cleared.
PCI Express Secondary Status Register[12] (received target abort) is set	PCI Express Secondary Status Interrupt Mask Register[2] (mask received target abort) is cleared.
PCI Express Secondary Status Register[13] (received master abort) is set	PCI Express Secondary Status Interrupt Mask Register[3] (mask received master abort) is cleared.
PCI Express Secondary Status Register[14] (signaled system error) is set.	PCI Express Secondary Status Interrupt Mask Register[4] (mask signaled system error) is cleared.
PCI Express Secondary Status Register[15] (detected parity error) is set	PCI Express Secondary Status Interrupt Mask Register[5] (mask detected parity error) is cleared.

Table 17-9. PCI Express Internal Controller Interrupt Sources (Continued)

Status Register Bit	Preconditions
PCI Express Slot Status Register[0] (attention button pressed) is set	PCI Express Slot Control Register[0] (attention button pressed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[1] (power fault detected) is set	PCI Express Slot Control Register[1] (power fault detected enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[2] (MRL sensor changed) is set	PCI Express Slot Control Register[2] (MRL sensor changed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[3] (presence detect changed) is set	PCI Express Slot Control Register[3] (presence detect changed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1–0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[4] (command completed) is set	PCI Express Slot Control Register[4] (command completed interrupt enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set.

17.3.4.3 Error Conditions

Table 17-10 describes specific error types and the action taken for various transaction types.

Table 17-10. 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.
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.

Table 17-10. Error Conditions (Continued)

Transaction Type	Error Type	Action
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	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.
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.

Table 17-10. Error Conditions (Continued)

Transaction Type	Error Type	Action
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.
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.
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	Internal platform memory write/read requests that do not hit either Memory Base/Limit or Pref Memory Base/Limit register.	Log the error in error detect register. The controller will drop the transaction.
Outbound request	Internal platform I/O write/read requests that do not hit either Memory Base/Limit or Pref Memory Base/Limit register.	Log the error in error detect register. The controller will drop the transaction.
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 response is the final response, then deallocate resource. If the response is not the final response, wait for all responses to come back before deallocate resource.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.

Table 17-10. Error Conditions (Continued)

Transaction Type	Error Type	Action
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.

17.3.5 Interrupts

Handling of INTx and message signaled interrupts (MSI) depends on whether the PCI Express controller is configured as an RC or EP device. As an RC device, the PCI Express controller responds to interrupts from the system. As a CP device, the controller can generate interrupt signals. **Table 17-11** and **Table 17-12** summarize the interrupt functionality for CP and RC devices, respectively.

Table 17-11. EP Device INTx and MSI Handling

Interrupt Action	EP Device
INTx Message Generation	
• Hardware	Not supported.
• Software	Not supported.
MSI Generation	
• Hardware:	When configured to handle this operation, the PCI Express controller can generate MSI transactions automatically to the RC. The PCI Express controller uses the GCR5[PEX_IRQ_OUT] bit (see **Chapter 8**, *General Configuration Registers* for details). to trigger the generation of the MSI. The remote RC must set up the MSI capability structure for all endpoints during system initialization by writing the appropriate values to the Message Address and Message Data registers and setting the MSIE bit in the MSI Message Control register. When configured properly, the PCI Express controller detects when the GCR5[PEX_IRQQ_OUT] bit sets (= 1) and 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).
• Software	The core must set up the MSI capability registers to enable MSI mode, and write the correct values to the MSI address and data register. Then, local software must read the MSI address in the MSI capability register and configure the outbound ATMU window to map the translated address to the MSI address (see **Chapter 8**, *General Configuration Registers* for details on programming the ATMU window). Software must determine the number of allocated messages in the MSI capability register and allocate 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.

Table 17-12. RC Device INTx and MSI Handling

Interrupt Action	RC Device
INTx Message	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 deasserted message, it asserts or deasserts an equivalent internal INTx signal (*inta*, *intb*, *intc*, or *intd*) to the EPIC in each core. If a PCI Express INTx interrupt is used, then the EPIC must be configured so that these interrupts are level-sensitive (see **Chapter 13**, *Interrupt Handling*).
MSI	An inbound MSI should be handled by using one of the virtual interrupts or virtual NMIs (see **Chapter 13**, *Interrupt Handling*). **Note:** It is the responsibility of the host software to configure the EP MSI capability registers such that an MSI cycle generated from the EP device generates an appropriate interrupt to the DSP cores.

17.3.6 Initial Credit Advertisement

To prevent overflowing of the receiver 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 17-13. Initial credit advertisement

Credit Type	Initial Credit Advertisement
PH (Memory Write, Message Write)	4
PD (Memory Write, Message Write)	(256/16) x 4 = 64
NPH (Memory Read, IO Read, Cfg Read, Cfg Write)	8
NPD (IO Write, Cfg Write)	2
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

17.3.7 Power Management

All device power states are supported with the exception of D3cold with Vaux. In addition, all link power states are supported with the exception of L2 states. 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 17-14. 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 traffics will be 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 via the PEX Power Management Command register.
D2	L0, L0s, L1	All outbound traffics are stalled. All inbound traffics will be 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 via the PEX Power Management Command register.
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffics are stalled. All inbound traffics will be 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 via 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 will restart.
D3cold	L3	Completely off.

17.3.8 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, the device can use one of the GPIO signals as an enable to an external tristate buffer to generate a $\overline{\text{WAKE}}$ signal, shown in **Figure 17-12**.

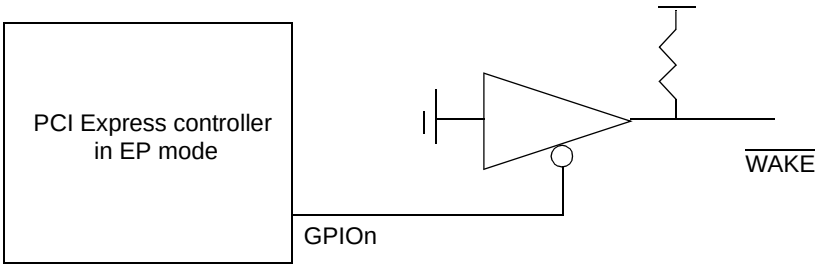


Figure 17-12. $\overline{\text{WAKE}}$ Generation Example

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.

17.3.9 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 non-sticky configuration register bits are cleared. Link training then occurs. The device can 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 cannot generate a hot reset condition; it can only detect a hot reset condition and initiate the appropriate clean-up procedure.

17.3.10 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.

Note: In EP mode, 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.

17.3.11 Initialization/Application Information

In normal boot mode (RCWHR[PRDY] = 0), the DSP cores are expected to boot and configure the device. During this time, the PCI Express interface will retry 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_CFG_READY register after which inbound PCI Express configuration transactions are accepted. Refer to **Section 17.4.1.9.20**, *Configuration Ready Register—0x4B0,* on page 17-119 for more information about the CFG_READY bit.

In PCI Express ready mode (RCWHR[PRDY] = 1), the PCI Express interface accepts all inbound PCI Express configuration transactions which allows an external host/RC to configure the device with DSP core intervention.

17.4 Programming Model

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 17.4.1**, *PCI Express Memory Mapped Registers,* on page 17-25, and its subsections.
- 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 17.4.1.6**, *PCI Express Configuration Space Access* and its subsections.

17.4.1 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 by the CCSR 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 17-15 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.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 17-15. PCI Express Memory-Mapped Register Map

Offset	Register	Access	Reset	Page
PCI Express Controller Memory-Mapped Registers—Block Base Address 0x0A000				
PCI Express Configuration Access Registers				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	**page 17-27**
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	**page 17-28**
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	**page 17-29**
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	**page 17-30**
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	**page 17-30**
PCI Express Power Management Event & Message Registers				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	**page 17-31**
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	**page 17-33**
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	**page 17-34**
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	**page 17-36**
PCI Express IP Block Revision Registers				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0101	**page 17-41**
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	**page 17-41**
PCI Express ATMU Registers				
Outbound Window 0 (Default)				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	**page 17-42**
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	**page 17-43**

Table 17-15. PCI Express Memory-Mapped Register Map (Continued)

Offset	Register	Access	Reset	Page
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	**page 17-45**
Outbound Window 1				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	**page 17-42**
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	**page 17-43**
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	**page 17-44**
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	**page 17-45**
Outbound Window 2				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	**page 17-42**
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	**page 17-43**
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	**page 17-44**
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	**page 17-45**
Outbound Window 3				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000	**page 17-42**
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	**page 17-43**
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000	**page 17-44**
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0004_4023	**page 17-45**
Outbound Window 4				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	**page 17-42**
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	**page 17-43**
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	**page 17-44**
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	**page 17-45**
Inbound Window 3				
0xDA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	**page 17-48**
0xDA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	**page 17-49**
0xDAC	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	**page 17-51**
0xDB0	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x20F4_4023	**page 17-52**
Inbound Window 2				
0xDC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	**page 17-48**
0xDC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	**page 17-49**
0xDCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	**page 17-51**
0xDD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	**page 17-52**
Inbound Window 1				
0xDE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	**page 17-48**
0xDE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	**page 17-49**
0xDF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x20F4_4023	**page 17-52**
PCI Express Error Management Registers				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	**page 17-54**
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	**page 17-56**
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	**page 17-58**

Table 17-15. PCI Express Memory-Mapped Register Map (Continued)

Offset	Register	Access	Reset	Page
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	**page 17-59**
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	**page 17-60**
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	**page 17-61**
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	**page 17-63**
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	**page 17-64**
0xE38–0xFFC	Reserved	—	—	

17.4.1.1 PCI Express Configuration Access Registers

17.4.1.1.1 PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

PEX_CONFIG_ADDR PCI Express Configuration Address Register Offset 0x00000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	—			EXTREGN				BUSN							
TYPE	R/W	R			RW											
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**
	DEVN				FUNCN				REGN						—	
TYPE	R/W														R	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCI Express configuration address register contains address information for accesses to PCI Express internal and external configuration registers. Both root complex (RC) and endpoint (EP) 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 fields of the PCI Express configuration address register are described in **Table 17-17**.

Table 17-16. PEX_CONFIG_ADDR Field Descriptions

Bits	Reset	Description	Settings
EN 31	0	**Enable** This bit allows a PCI Express configuration access when PEX_CONFIG_DATA is accessed.	0 Writing to PEX_CONFIG_DATA has no effect and reading PEX_CONFIG_DATA returns unknown data. 1 PCI Express configuration access allowed when PEX_CONFIG_DATA is accessed.
— 30–28	0	Reserved. Write to zero for future compatibility.	
EXTREGN 27–24	0	**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 17-16. PEX_CONFIG_ADDR Field Descriptions (Continued)

Bits	Reset	Description	Settings
BUSN 23–16	0	**BUS Number** PCI bus number to access.	
DEVN 15–11	0	**Device Number** Device number to access on specified bus.	
FUNCN 10–8	0	**Function Number** Function to access within specified device.	
REGN 7–2	0	**Register Number** 32-bit register to access within specified device	
— 1–0	0	Reserved. Write to zero for future compatibility.	

Table 17-17. PEX_CONFIG_ADDR Field Descriptions

Bits	Name	Description
1–3	—	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).
8–15	BUSN	Bus number. PCI bus number to access
16–20	DEVN	Device number. Device number to access on specified bus
21–23	FUNCN	Function number. Function to access within specified device
24–29	REGN	

17.4.1.1.2 PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The PCI Express configuration data register, show in **Figure 17-1**, 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 17.3.2.2**, *Byte Order for Configuration Transactions*, on page 17-9 for more information.

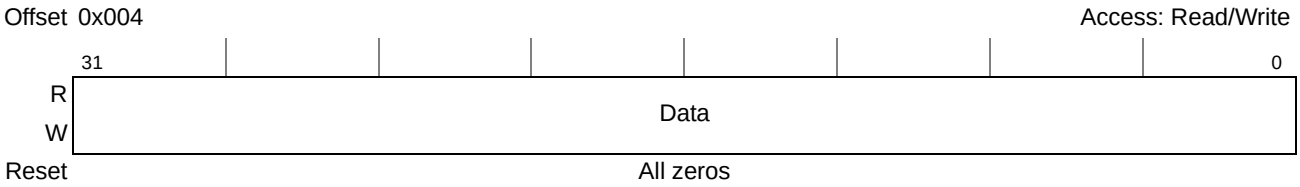


Figure 17-13. PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The fields of the PCI Express configuration data register are described in **Table 17-18**.

Table 17-18. PEX_CONFIG_DATA Field Descriptions

Bits	Name	Description
310	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).

17.4.1.1.3 PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The PCI Express outbound completion timeout register, shown in **Figure 17-2**, 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. The outbound completion timeout register (see <Cross Refs>Figure 17-14) is a programmable register that represents 1/8 timeout period of a response from PCI Express. The logic uses this register's programmed value as an input to a running counter. This counter gets loaded with the timeout counter value every time a non-posted request is scheduled to PCI Express. While waiting for the response to come back, the counter gets decremented. If the response did not come back when the timer expires, then a timeout error has occurred. The specification requires a minimum of 10 ms and a maximum of 50 ms before timeout.

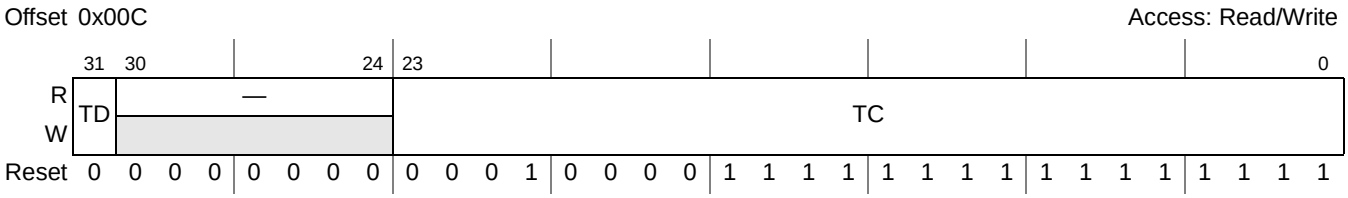


Figure 17-14. PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The fields of the PCI Express outbound completion timeout register are described in **Table 17-19**.

Table 17-19. PEX_OTB_CPL_TOR Field Descriptions

Bits	Name	Description
31	TD	Timeout disable. This bit controls the enabling/disabling of the timeout function. 0 Enable completion timeout 1 Disable completion timeout
30–24	—	Reserved
23–0	TC	Timeout counter. This is the value that is used to load the response counter of the completion timeout. One TC unit is 24 ns at 333 MHz and 30 ns at 260 MHz. Timeout period based on different TC settings: 0x00_0000 Reserved 0x10_FFFF 26.72 ms at 333.33 MHz platform clock 0xFF_FFFF 446.94 ms at 333.33 MHz platform clock

17.4.1.1.4 PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The PCI Express configuration retry timeout register, shown in **Figure 17-3**, contains the maximum time period during which retries of configuration transactions which resulted in a CRS response occur.

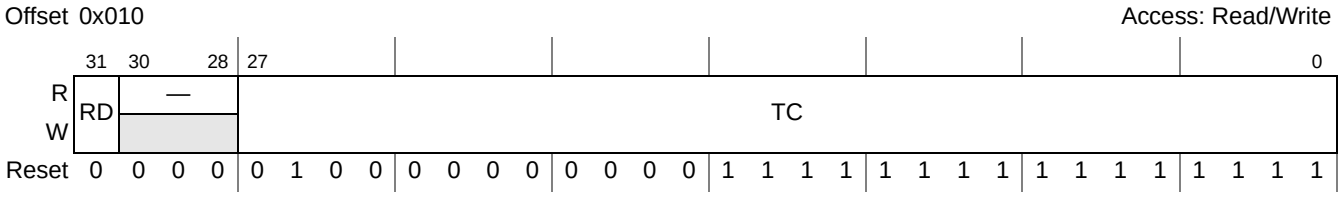


Figure 17-15. PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The fields of the PCI Express configuration retry timeout register are described in **Table 17-20**.

Table 17-20. PEX_CONF_RTY_TOR Field Descriptions

Bits	Name	Description
31	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.
30–28	—	Reserved
27–0	TC	Timeout counter. This is the value that is used to load the CRS response counter. One TC unit is 24 ns at 333 MHz and 30 ns at 260 MHz. Timeout period based on different TC settings: 0x000_0000 Reserved 0x400_FFFF 1.612 seconds at 333.33 MHz platform clock 0xFFF_FFFF 7.12 seconds at 333.33 MHz platform clock

17.4.1.1.5 PCI Express Configuration Register (PEX_CONFIG)

The PCI Express configuration register, shown in **Figure 17-4**, contains various control switches for the controller.

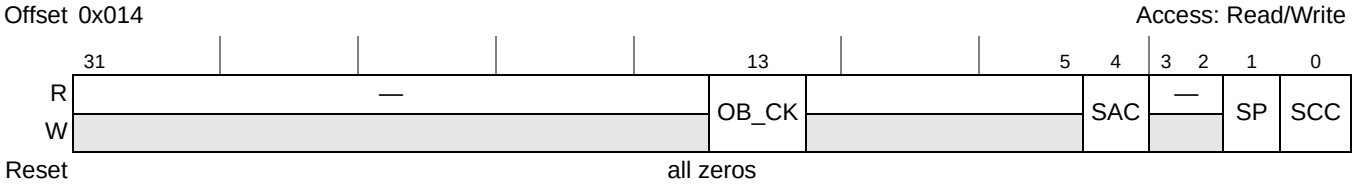


Figure 17-16. PCI Express Configuration Register (PEX_CONFIG)

The fields of the PCI Express configuration register are described in **Table 17-21**.

Table 17-21. PEX_CONFIG Field Descriptions

Bits	Name	Description
31–14	—	Reserved
13	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.
12–5	—	Reserved
4	SAC	Sense ASPM Control. This bit controls the default value of ASPM of PEX Link Control Register's bit 0. See **Section 17.4.1.8.11**, *PCI Express Link Control Register—0x5C,* on page 17-96 for more information.
3–2	—	Reserved
1	SP	Slot Present. This bit controls the default value of the PCI Express capabilities register [slot] bit. See **Section 17.4.1.8.6**, *PCI Express Capabilities Register—0x4E,* on page 17-93 for more information.
0	SCC	Slot Clock Configuration. This bit controls the default value of the PCI Express link status register [SCC] bit. See **Section 17.4.1.8.12**, *PCI Express Link Status Register—0x5E,* on page 17-97 for more information.

17.4.1.2 PCI Express Power Management Event and Message Registers

17.4.1.2.1 PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The PCI Express PME and message detect register, shown in **Figure 17-5**, logs inbound messages and PME events that are detected by the PCI Express controller. This register is a write-1-to-clear type register.

Offset 0x020 Access: w1c

	31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—	PTO	PTAT	ENL23	EXL23	—	HRD	LDD	—	AION	AIB	AIOF	PION	PIB	PIOF	ABP	
W		w1c	w1c	w1c	w1c		w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros																

Figure 17-17. 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 17-22**.

Table 17-22. PEX_PME_MES_DR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15	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
14	PTAT	PME to ack time-out. This bit indicates the detection of a PME_to_ack time-out condition. This bit is only valid in RC mode. 1 A PME_TO_Ack time-out condition is detected 0 No PME_TO_Ack time-out condition detected
13	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

Table 17-22. PEX_PME_MES_DR Field Descriptions (Continued)

Bits	Name	Description
12	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
11	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
10	HRD	Hot reset detected. This bit indicates that the PCI Express controller has detected a hot reset condition on the link. The controller will be reset and will clean up all outstanding transactions. Link retraining will take 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
9	LDD	Link down detected. This bit indicates that a link down condition has been detected. The controller will be reset and then will clean up all outstanding transactions. Link retraining will take place once the controller has cleaned itself up. 1 Link down has been detected 0 Link down not detected
8–7	—	Reserved
6	AION	Attention indicator on. This bit indicates the detection of an Attention_Indicator_On message. This bit is only valid in EP mode. 1 Attention indicator on message is detected 0 No attention indicator on message detected
5	AIB	Attention indicator blink. This bit indicates the detection of an Attention_Indicator_Blink message. This bit is only valid in EP mode. 1 Attention indicator blink message is detected 0 No attention indicator blink message detected
4	AIOF	Attention indicator off. This bit indicates the detection of an Attention_Indicator_Off message. This bit is only valid in EP mode. 1 Attention indicator off message is detected 0 No attention indicator off message detected
3	PION	Power indicator on. This bit indicates the detection of a Power_Indicator_On message. This bit is only valid in EP mode. 1 Power indicator on message is detected 0 No power indicator on message detected
2	PIB	Power indicator blink. This bit indicates the detection of an Power_Indicator_Blink message. This bit is only valid in EP mode. 1 Power indicator blink message is detected 0 No power indicator blink message detected
1	PIOF	Power indicator off. This bit indicates the detection of an Power_Indicator_Off message. This bit is only valid in EP mode. 1 Power indicator off message is detected 0 No power indicator off message detected
0	ABP	Attention button pressed. This bit indicates the detection of an Attention_Button_Pressed message. This bit is only valid in EP mode. 1 Attention button press message is detected 0 No attention button press message detected

17.4.1.2.2 PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The PCI Express PME and message disable register, shown in **Figure 17-6**, when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register.

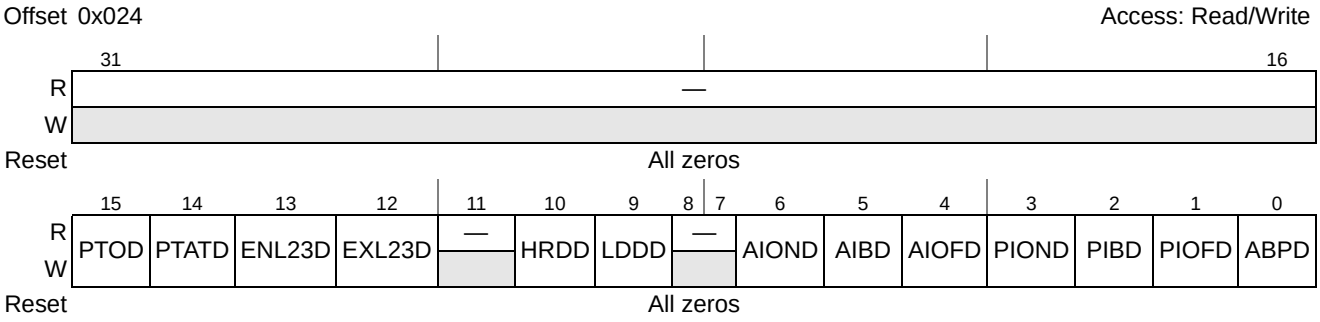


Figure 17-18. PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The fields of the PCI Express PME and message disable register are described in **Table 17-23**.

Table 17-23. PEX_PME_MES_DISR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15	PTOD	PME turn off disable. When set will disable the setting of PEX_PME_MES_DR[PTO] bit. 1 Disable PME_Turn_Off_message detection 0 Enable PME_Turn_Off message detection
14	PTATD	PME to ack time-out disable. When set will disable the setting of PEX_PME_MES_DR[PTAT] bit. 1 Disable PME_TO_Ack time-out detection 0 Enable PME_TO_Ack time-out detection
13	ENL23D	Entered_L2/L3 ready disable. When set will disable 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
12	EXL23D	Exited_L2/L3 ready disable. When set will disable 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
11	—	Reserved
10	HRDD	Hot reset detected disable. When set will disable the setting of PEX_PME_MES_DR[HRD] bit. 1 Disable hot reset state detection 0 Enable hot reset state detection
9	LDDD	Link down detected disable. When set will disable the setting of PEX_PME_MES_DR[LDD] bit. 1 Disable link down state detection 0 Enable link down state detection
8–7	—	Reserved
6	AIOND	Attention indicator on disable. When set will disable the setting of PEX_PME_MES_DR[AION] bit. 1 Disable attention indicator on message detection 0 Enable attention indicator on message detection
5	AIBD	Attention indicator blink disable. When set will disable the setting of PEX_PME_MES_DR[AIB] bit. 1 Disable attention indicator blink message detection 0 Enable attention indicator blink message detection
4	AIOFD	Attention indicator off disable. When set will disable the setting of PEX_PME_MES_DR[AIOF] bit. 1 Disable attention indicator off message detection 0 Enable attention indicator off message detection

Table 17-23. PEX_PME_MES_DISR Field Descriptions (Continued)

Bits	Name	Description
3	PIOND	Power indicator on disable. When set will disable the setting of PEX_PME_MES_DR[PION] bit. 1 Disable power indicator on message detection 0 Enable power indicator on message detection
2	PIBD	Power indicator blink disable. When set will disable the setting of PEX_PME_MES_DR[PIB] bit. 1 Disable power indicator blink message detection 0 Enable power indicator blink message detection
1	PIOFD	Power indicator off disable. When set will disable the setting of PEX_PME_MES_DR[PIOF] bit. 1 Disable power indicator off message detection 0 Enable power indicator off message detection
0	ABPD	Attention button pressed disable. When set will disable the setting of PEX_PME_MES_DR[ABP] bit. 1 Disable attention button press message detection 0 Enable attention button press message detection

17.4.1.2.3 PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

The PCI Express PME and message interrupt enable register, shown in **Figure 17-7**, 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.

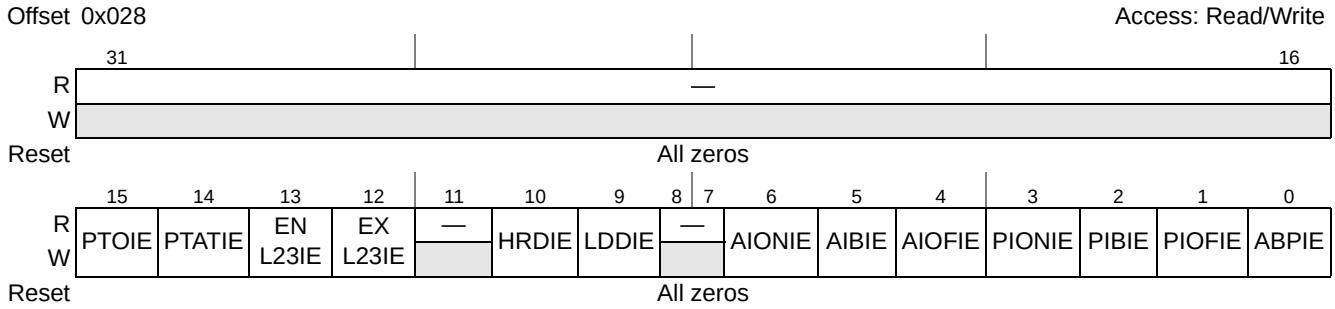


Figure 17-19. PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

Table 17-24 shows the fields of the PCI Express PME and message interrupt enable register.

Table 17-24. PEX_PME_MES_IER Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15	PTOIE	PME turn off interrupt enable. When set and PEX_PME_MES_DR[PTO]=1 will generate an interrupt. 1 Enable PME_Turn_Off_message interrupt generation 0 Disable PME_Turn_Off message interrupt generation
14	PTATIE	PME To ack time-out interrupt enable. When set and PEX_PME_MES_DR[PTAT]=1 will generate an interrupt. 1 Enable PME_TO_Ack time-out interrupt generation 0 Disable PME_TO_Ack time-out interrupt generation
13	ENL23IE	Entered L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[ENL23]=1 will generate an interrupt. 1 Enable Entered_L2/L3 ready state interrupt generation 0 Disable Entered_L2/L3 ready state interrupt generation

Table 17-24. PEX_PME_MES_IER Field Descriptions (Continued)

Bits	Name	Description
12	EXL23IE	Exited L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[EXL23]=1 will generate an interrupt. 1 Enable Exited_L2/L3 ready state interrupt generation 0 Disable Exited_L2/L3 ready state interrupt generation
11	—	Reserved
10	HRDIE	Hot reset detected interrupt enable. When set and PEX_PME_MES_DR[HRD]=1 will generate an interrupt. 1 Enable hot reset state interrupt generation 0 Disable hot reset state interrupt generation
9	LDDIE	Link down detected interrupt enable. When set and PEX_PME_MES_DR[LDD]=1 will generate an interrupt. 1 Enable link down state interrupt generation 0 Disable link down state interrupt generation
8–7	—	Reserved
6	AIONIE	Attention indicator on interrupt enable. When set and PEX_PME_MES_DR[AION]=1 will generate an interrupt. 1 Enable attention indicator on message interrupt generation 0 Disable attention indicator on message interrupt generation
5	AIBIE	Attention indicator blink interrupt enable. When set and PEX_PME_MES_DR[AIB]=1 will generate an interrupt. 1 Enable attention indicator blink message interrupt generation 0 Disable attention indicator blink message interrupt generation
4	AIOFIE	Attention indicator off interrupt enable. When set and PEX_PME_MES_DR[AIOF]=1 will generate an interrupt. 1 Enable attention indicator off message interrupt generation 0 Disable attention indicator off message interrupt generation
3	PIONIE	Power indicator on interrupt enable. When set and PEX_PME_MES_DR[PION]=1 will generate an interrupt. 1 Enable power indicator on message interrupt generation 0 Disable power indicator on message interrupt generation
2	PIBIE	Power indicator blink interrupt enable. When set and PEX_PME_MES_DR[PIB]=1 will generate an interrupt. 1 Enable power indicator blink message interrupt generation 0 Disable power indicator blink message interrupt generation
1	PIOFIE	Power indicator off interrupt enable. When set and PEX_PME_MES_DR[PIOF]=1 will generate an interrupt. 1 Enable power indicator off message interrupt generation 0 Disable power indicator off message interrupt generation
0	ABPIE	Attention button pressed interrupt enable. When set and PEX_PME_MES_DR[ABP]=1 will generate an interrupt. 1 Enable attention button press message interrupt generation 0 Disable attention button press message interrupt generation

17.4.1.2.4 PCI Express Power Management Command Register (PEX_PMCR)

The PCI Express power management command register, shown in **Figure 17-8**, provides software a mechanism to allow the PCI Express controller to get back to L0 link state.

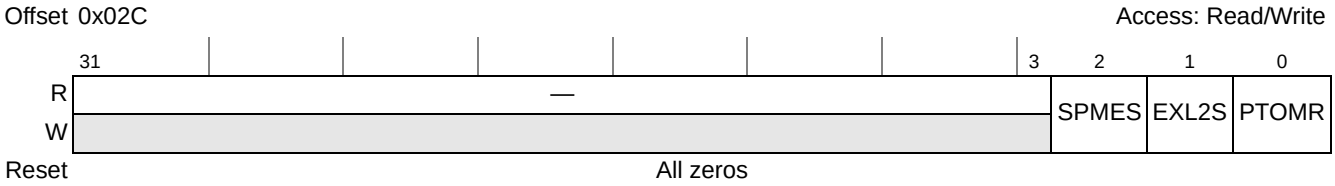


Figure 17-20. PCI Express Power Management Command Register (PEX_PMCR)

The fields of the PCI Express power management command register are described in **Table 17-25**.

Table 17-25. PEX_PMCR Field Descriptions

Bits	Name	Description
31–3	—	Reserved
2	SPMES	Set PME status. This will set the PME status bit and if PME is enabled (see **Section 17.4.1.8.3**, *PCI Express Power Management Status and Control Register—0x48,* on page 17-92 for more information) it will transmit a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.
1	EXL2S	Exit L2 state. When set will exit 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 will be set. This bit should not be used when in EP mode.
0	PTOMR	PME_Turn_Off message request. When set will broadcast a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing

17.4.1.2.5 PCI Express Link Width Control Register (PEX_LWCR)

The PCI Express link width control register, shown in **Figure 17-21**, provides software a mechanism to dynamically changing the link width.

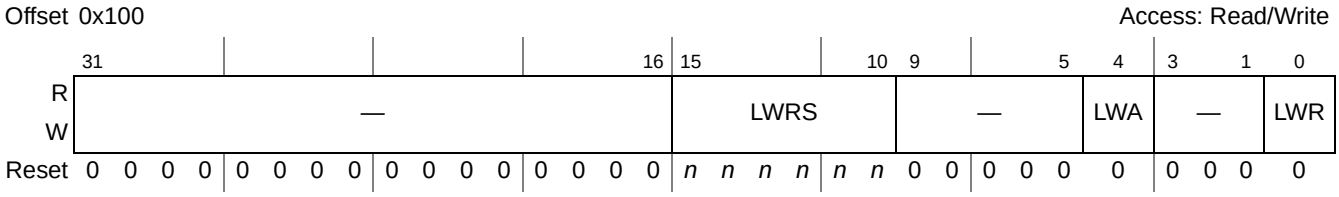


Figure 17-21. PCI Express Link Width Control Register (PEX_LWCR)

The fields of the PCI Express link width control register are described in **Table 17-26**.

Table 17-26. PEX_LWCR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15-10	LWRS	Link width request size. These bits indicate the size of the link width request. Software should first read the PEX_LWSR[LD] bits to determine what are the valid lanes to request before initializing this field. 000001 - x1 link 000010 - x2 link 000100 - x4 link All other encodings are reserved and should not be used.
9–5	—	Reserved
4	LWA	Link width auto. This bit is set by software.
3–1	—	Reserved
0	LWR	Link width request. This bit when set by software will initiate a change in link width as indicated by LWRS. Once the link width operation finishes, this bit is cleared by hardware.

17.4.1.2.6 PCI Express Link Width Status Register (PEX_LWSR)

The PCI Express link width status register, shown in **Figure 17-22**, provides software a mechanism to dynamically changing the link width.

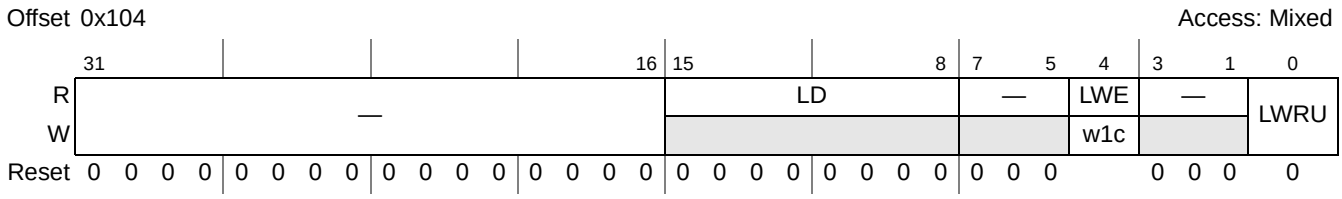


Figure 17-22. PCI Express Link Width Status Register (PEX_LWCR)

The fields of the PCI Express link width status register are described in **Table 17-27**.

Table 17-27. PEX_LWSR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15-8	LD	Lane detected. Each bit indicates a corresponding lane was detected when coming out of the Detect state. For example, a value of 00000001 means only one lane was detected; a value of 00000011 means 2 lanes were detected.
7–5	—	Reserved
4	LWE	Link width error. An error has been detected when making a link width change request. 0 no error 1 error has been detected. The following conditions will cause an error to be detected. • PEX_LWCR[LWR]=1 and PEX_LWCR[LWA]=1 and Hardware Autonomous Width Disable bit is set in Link Control 2 register • PEX_LWCR[LWR] = 1 and PEX_LWCR[LWRS] > current link width and not upconfiguration capable • PEX_LWCR[LWR] = 1 and PEX_LWCR[LWRS] width is not to available lanes • PEX_PME_MES_DR[LWD]=1 and current width != PEX_LWCR[LWRS]
3–1	—	Reserved
0	LWU	Link width upconfiguration capable. It means that the link is capable of a size up request.

17.4.1.2.7 PCI Express Link Speed Control Register (PEX_LSCR)

The PCI Express link speed control register, shown in **Figure 17-23**, provides software a mechanism to dynamically changing the link speed.

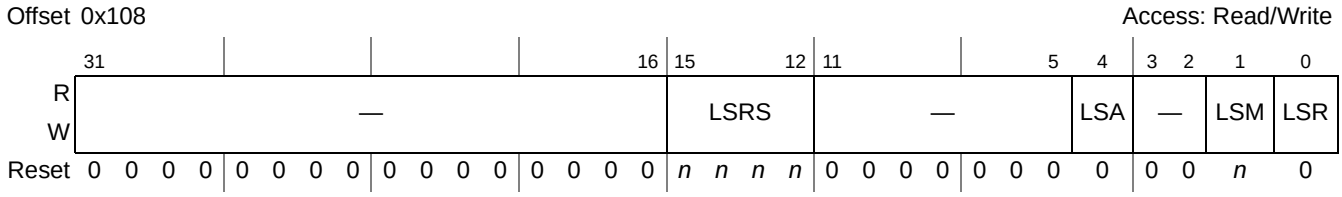


Figure 17-23. PCI Express Link Speed Control Register (PEX_LSCR)

The fields of the PCI Express link speed control register are described in **Table 17-28**.

Table 17-28. PEX_LSCR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15-12	LSRS	Link speed request size. These bits indicate the size of the link speed request. Software should first determine the link partner's speed by reading PEX_LSSR[LPAS] bits before initializing these bits. 0001 - 2.5 Gb/s speed 0010 - 5.0 Gb/s speed All other encodings are reserved and should not be used.
11–5	—	Reserved
4	LSA	Link speed auto. This bit is set by software. It is gated with the Hardware Autonomous Speed Disable bit.
3-2	—	Reserved
1	LSM	Link speed mask. This bit when set indicates that 5.0 Gb/s speed is not supported.
0	LSR	Link speed request. This bit when set by software will initiate a change in link speed as indicated by LSRS. Once the link speed operation finishes, this bit is cleared by hardware.

17.4.1.2.8 PCI Express Link Speed Status Register (PEX_LSSR)

The PCI Express link speed request status register, shown in **Figure 17-20**, provides software status of the link speed request.

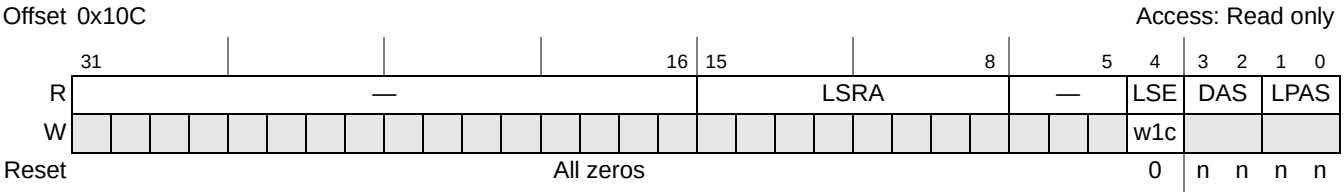


Figure 17-24. PCI Express Link Speed Status Register (PEX_LSSR)

The fields of the PCI Express link speed status register are described in 17-25.

Table 17-29. PEX_LSSR Field Descriptions

Bits	Name	Description
31–16	—	Reserved
15–8	LSRA	Link speed request attempt. These bits indicate the number of times that the link speed has been changed (or attempt to change)
7–5	—	Reserved
4	LSE	Link speed error. When set indicates that there was an error with the link speed request. 0 no error 1 error has been detected. The following conditions will cause an error to be detected. • PEX_LSCR[LSR]=1 and PEX_LSCR[LSA]=1 and Hardware Autonomous Speed Disable bit is set in Link Control 2 register • PEX_LSCR[LSR] = 1 and PEX_LWCR[LSRS] = 4’b0010 and PEX_LSSR[LPAS] = 2’b01 • PEX_PMS_MES_DR[LSD]=1 and current speed != PEX_LWCR[LSRS]
3–2	DAS	Device advertised speed. 00 Reserved 01 2.5 Gb/s 10 5.0 Gb/s 11 Reserved
1–0	LPAS	Link partner advertised speed. Its default value is set by the link partner’s speed. 00 Reserved 01 2.5 Gb/s 10 5.0 Gb/s 11 Reserved

17.4.1.3 PCI Express IP Block Revision Registers

17.4.1.3.1 IP Block Revision Register 1 (PEX_IP_BLK_REV1)

The IP block revision register 1 is shown in **Figure 17-9**.

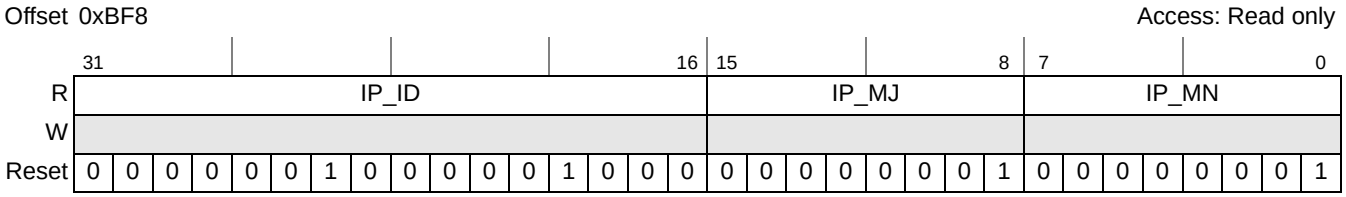


Figure 17-25. IP Block Revision Register 1

Table 17-30 contains descriptions of the fields of the IP block revision register 1. This read-only register indicates the identification and revision of the source IP used in the device design.

Table 17-30. PCI Express IP Block Revision Register 1 Field Descriptions

Bits	Name	Description
31–16	IP_ID	Block ID
15–8	IP_MJ	Block Major Revision
7–0	IP_MN	Block Minor Revision

17.4.1.3.2 IP Block Revision Register 2 (PEX_IP_BLK_REV2)

The IP block revision register 2 is shown in **Figure 17-12**.

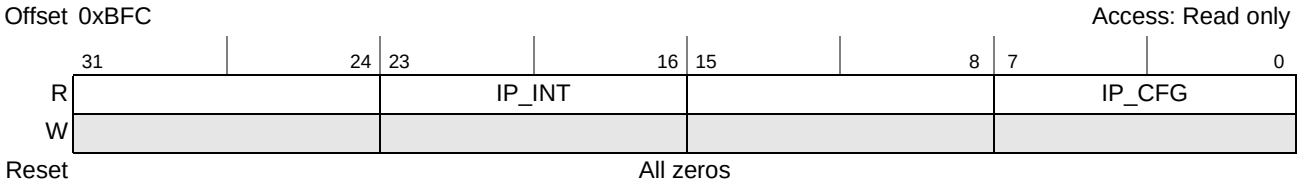


Figure 17-26. IP Block Revision Register 2

Table 17-31 contains descriptions of the fields of the IP block revision register 2. This read-only register indicates the identification and revision of the source IP used in the device design.

Table 17-31. PCI Express IP Block Revision Register 2 Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–16	IP_INT	Block integration option
15–8	—	Reserved
7–0	IP_CFG	Block configuration option

17.4.1.4 PCI Express ATMU Registers

17.4.1.4.1 PCI Express Outbound ATMU Registers

The outbound address translation windows must be aligned based on the granularity selected by the size fields. Outbound window misses use the default outbound register set (outbound ATMU window 0). Overlapping outbound windows 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.

Note that in RC mode, there is no checking on whether the translated address actually hits into the memory base/limit range. It will just pass it through as is.

Figure 17-27 shows the outbound transaction flow.

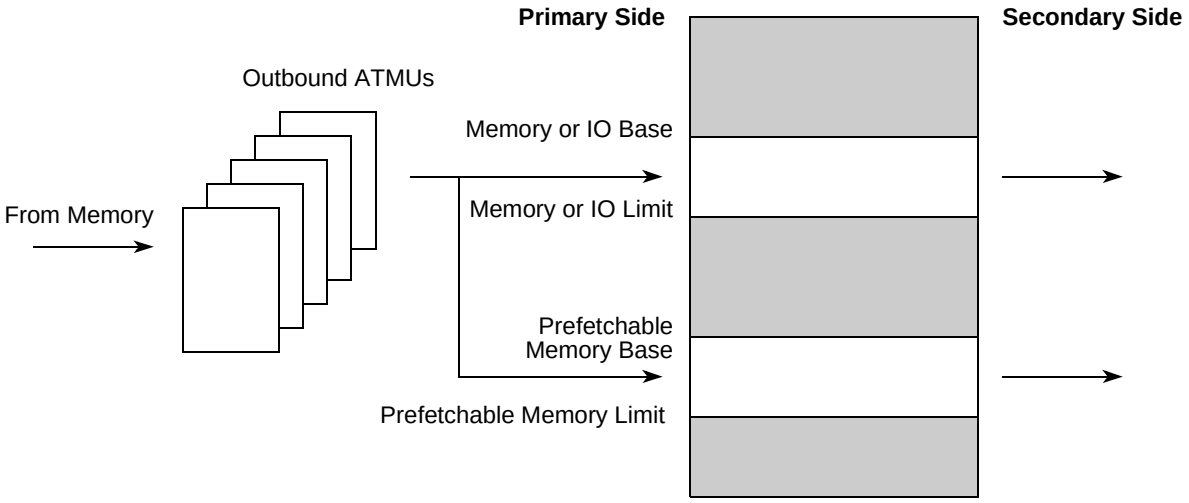


Figure 17-27. RC Outbound Transaction Flow

17.4.1.4.2 PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

The PCI Express outbound translation address registers, shown in **Figure 17-28**, 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.

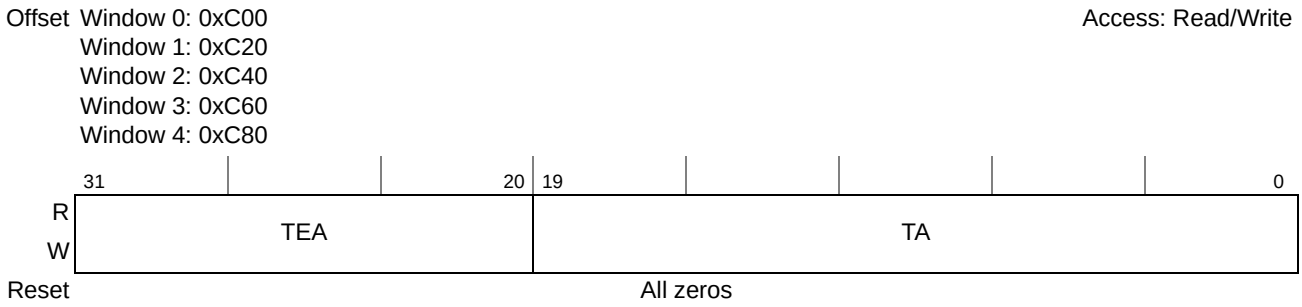


Figure 17-28. PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

Table 17-32 describes the fields of the PCI Express outbound translation address registers.

Table 17-32. PEXOTAR*n* Field Descriptions

Bits	Name	Description
31–20	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).
19–0	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].

17.4.1.4.3 PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

The PCI Express outbound translation extended address registers, shown in **Figure 17-29**, contain the most-significant bits of a 64 bit translation address.

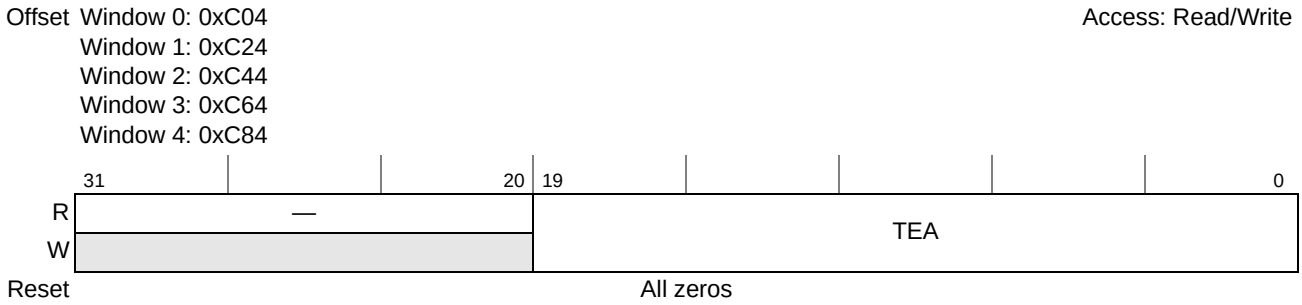


Figure 17-29. PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

Table 17-33 describes the fields of the PCI Express outbound translation extended address registers.

Table 17-33. PCI Express Outbound Extended Address Translation Register *n* Field Descriptions

Bits	Name	Description
31–20	—	Reserved
19–0	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].

17.4.1.4.4 PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

The PCI Express outbound window base address registers, shown in **Figure 17-30**, 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.

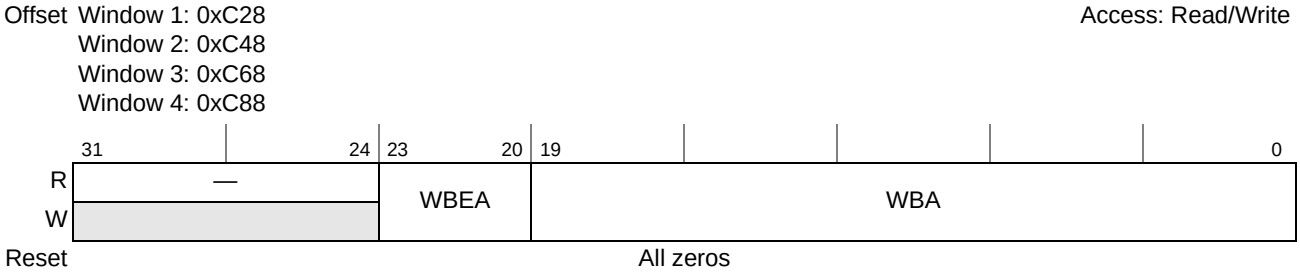


Figure 17-30. PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

Table 17-34 describes the fields of the PCI Express outbound window base address registers.

Table 17-34. PCI Express Outbound Window Base Address Register *n* Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–20	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 [0:3]. (where 0 is the msb of the internal platform address)
19–0	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 [4:23].

17.4.1.4.5 PCI Express Outbound Window Attributes Registers (PEXOWAR*n*)

The PCI Express outbound window attributes registers, shown in **Figure 17-31** and **Figure 17-32**, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed. **Figure 17-31** shows the outbound window attributes register 0 (PEXOWAR0).

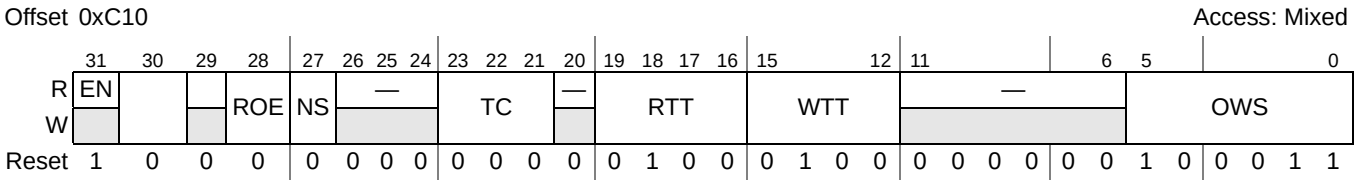


Figure 17-31. PCI Express Outbound Window Attributes Register 0 (PEXOWAR0)

Figure 17-32 shows the PCI Express outbound window attributes registers 1–4 (PEXOWAR*n*).

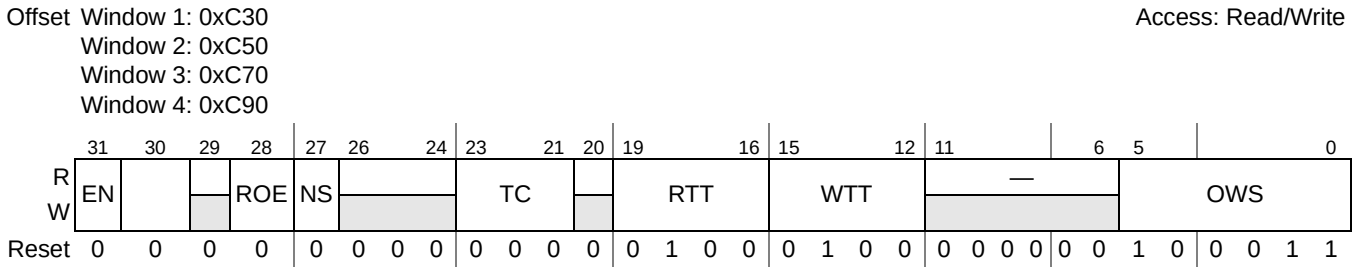


Figure 17-32. PCI Express Outbound Window Attributes Registers 1–4 (PEXOWAR*n*)

Table 17-35 describes the fields of the PCI Express outbound window attributes registers.

Table 17-35. PEXOWAR*n* Field Descriptions

Bits	Name	Description
31	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
30–29	—	Reserved
28	ROE	Relaxed ordering enable. This bit when set and the PCI Express device control register[Enable Relaxed] bit is set will enable the Relaxed Ordering bit for the packet. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
27	NS	No snoop enable. This bit when set and the PCI Express device control register[Enable No Snoop] bit is set will enable the no snoop bit for the packet. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
26–24	—	Reserved

Table 17-35. PEXOWAR*n* Field Descriptions (Continued)

Bits	Name	Description
23–21	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.
20	—	Reserved
19–16	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
15–12	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
11–6	—	Reserved
5–0	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 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window will be 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 100100 Reserved ... 111111 Reserved

17.4.1.4.6 PCI Express Inbound ATMU Registers

There are differences between RC and EP implementations of inbound ATMU registers as described in the following sections.

17.4.1.4.7 EP Inbound ATMU Implementation

All base address registers (BARs) reside in the PCI Express type 0 configuration header space which is accessible via 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 1-Mbyte window used for inbound memory transactions that access memory-mapped registers.
- Inbound window BAR1 at configuration address 0x14 (32-bit)
- Inbound window BAR2 at configuration address 0x18-0x1c (64-bit)
- Inbound window BAR3 at configuration address 0x20-0x24 (64-bit)

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 in the memory-mapped register set for the translation. If the transaction hits multiple BARs, then the lowest-numbered BAR will be used.

17.4.1.4.8 RC Inbound ATMU Implementation

In RC mode, the PEXIWBAR[1–3] registers reside outside of the type 1 header; PEXIWBAR0 is the only inbound BAR that resides in the Type 1 header (at offset 0x10).

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 a UR completion will be returned for non-posted transactions. All posted transactions with no BAR hit are ignored.

Figure 17-33 shows the inbound transaction flow in RC mode.

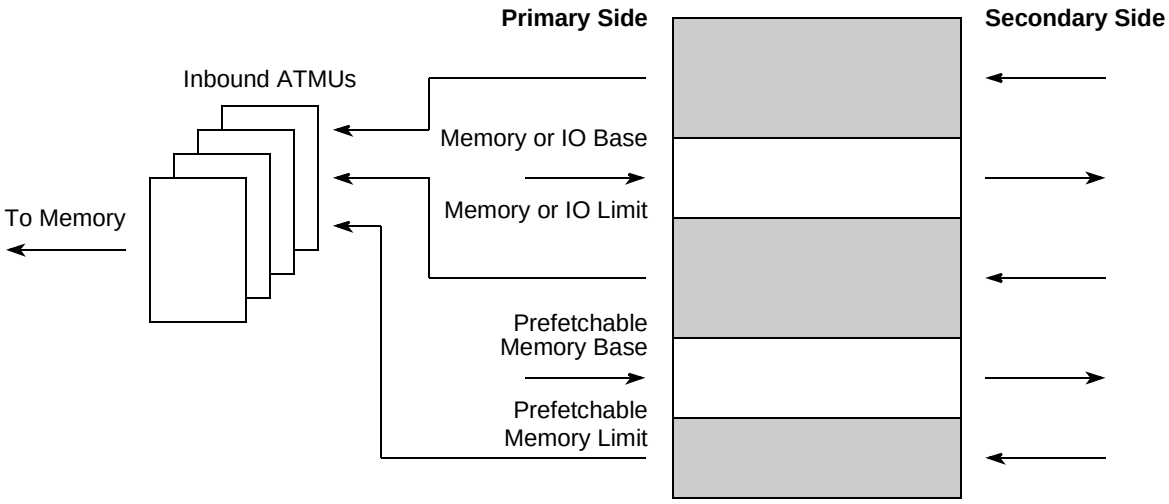


Figure 17-33. RC Inbound Transaction Flow

17.4.1.4.9 PCI Express Inbound Translation Address Registers (PEXITAR*n*)

The PCI Express inbound translation address registers, shown in **Figure 17-34**, contain the translated internal platform address to be used. Note that PEXITAR0 does not exist in the memory-mapped space; it is a fixed 1-Mbyte translation to the internal configuration (CCSR) space.

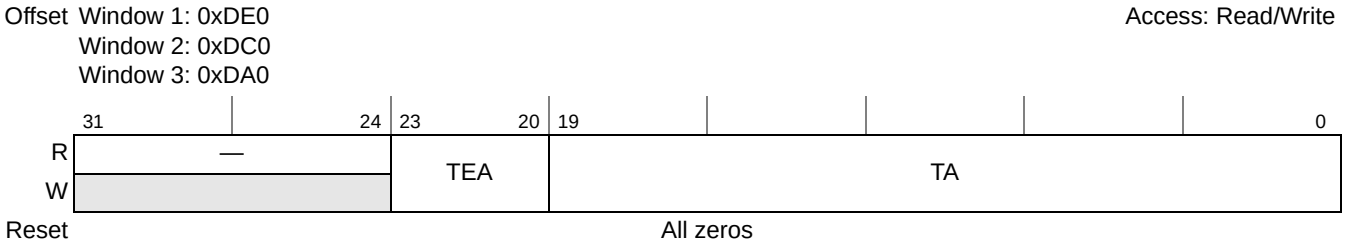


Figure 17-34. PCI Express Inbound Translation Address Registers (PEXITAR*n*)

Table 17-36 describes the fields of the PCI Express inbound translation address registers.

Table 17-36. PCI Express Inbound Translation Address Registers Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–20	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 [0:3] where bit 0 is the msb of the internal platform address.
19–0	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. This corresponds to internal platform address bits [4:23].

17.4.1.4.10 PCI Express Inbound Window Base Address Register 1 (PEXIWBAR1)

PEXIWBAR1 PCI Express Inbound Window Base Address Register 1 Offset 0xDE8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												WBA			
TYPE	R												R/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**
	WBA															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCI Express inbound window base address register 1 selects the base address for the window that is translated to an alternate target address space. In root complex (RC) mode, addresses for inbound transactions are compared to the PEXIWBAR 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 effect. 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.

Table 17-37 describes the fields of the PCI Express inbound window base address registers.

Table 17-37. PEXIWBAR*x* Field Descriptions

Bit	Reset	Description
— 31–20	0	Reserved. Write to zero for future compatibility.
WBA 19–0	0	**Window Base Address** Source address that is the starting-point for the inbound translation window. The window must be aligned on the basis of the size selected in the window size bits. This corresponds to PCI Express address bits 31–12.

17.4.1.4.11 PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

PEXIWBAR2 PCI Express Inbound Window Base Address Registers 2–3 Offset 0xDC8
PEXIWBAR3 Offset 0xDA8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WBEA												WBA			
TYPE	R/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**
	WBA															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCI Express inbound window base address registers select the base address for the windows that 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 effect. 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.

Table 17-38 describes the fields of the PCI Express inbound window base address registers.

Table 17-38. PEXIWBAR*x* Field Descriptions

Bit	Reset	Description
WBEA 31–20	0	**Window Base Extended Address** This field corresponds to PCI Express address bits [43–32].
WBA 19–0	0	**Window Base Address** Source address that is the starting-point for the inbound translation window. The window must be aligned on the basis of the size selected in the window size bits. This corresponds to PCI Express address bits 31–12.

17.4.1.4.12 PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

PEXIWBEAR2 PCI Express Inbound Window Base Extended Address Registers 2–3 Offset 0xDCC
PEXIWBEAR3 Offset 0xDAC

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—												WBEA			
TYPE	R												R/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**
	WBEA															
TYPE	R/W															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCI Express inbound window base extended address registers contain the most-significant bits of a 64 bit base address. **Table 17-39** describes the fields of the PCI Express inbound window base extended address registers.

Table 17-39. PEXIWBAR*x* Field Descriptions

Bit	Reset	Description
— 31–20	0	Reserved. Write to zero for future compatibility.
WBEA 19–0	0	**Window Base Extended Address** This field corresponds to PCI Express address bits [63:44]

17.4.1.4.13 PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

PEXIWAR1	PCI Express Inbound	Offset 0xDF0
PEXIWAR2	Window Attributes	Offset 0xDD0
PEXIWAR3	Registers 1–3	Offset 0xDB0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EN	—	PF	—					TRGT				RTT			
TYPE	R/W			R					R/W							
RESET	0	0	1	0	0	0	0	0	1	1	1	1	0	1	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	WTT			—					IWS							
TYPE	R/W			R					R/W							
RESET	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

The PCI Express inbound window attributes registers define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed. **Table 17-40** describes the fields of the PCI Express inbound window attributes registers.

Table 17-40. PEXIWAR*x* Field Descriptions

Bit	Description	Settings
EN 31	**Enable Address Translation** Set to 1 and read-only for default window 0.	0 Disable inbound window translation. 1 Enable inbound window translation.
— 30	Reserved. Write to zero for future compatibility.	
PW 29	**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 prefetchable bit in EP mode.	0 Not prefetchable 1 Prefetchable
— 28–24	Reserved. Write to zero for future compatibility.	
TRGT 23–20	**Target Interface** This field selects the interface to which the inbound transaction is sent. See **Section 15.1** for information about the OCN-to-MBus bridges.	0000 OCN-to-MBus Bridge 0 1111 OCN-to-MBus Bridge 1 All others reserved.
RTT 19–16	**Read Transaction Type** Transaction type to run on the local memory if the access is a read.	Not a local memory space access: 0100 Read. All others reserved.
WTT 15–12	**Write Transaction Type** Transaction type to run on local memory if access is a write.	Not a local memory space access: 0100 Write All others reserved.

Table 17-40. PEXIWAR*x* Field Descriptions (Continued)

Bit	Description	Settings
— 11–6	Reserved. Write to zero for future compatibility.	
IWS 5–0	**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.	001011 4-Kbyte window size 001100 8-Kbyte window size to in 4-Kbyte increments 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 All others reserved.

17.4.1.5 PCI Express Error Management Registers

17.4.1.5.1 PCI Express Error Detect Register (PEX_ERR_DR)

The PCI Express error detect register, shown in **Figure 17-35**, 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 will set the appropriate error bits in the error detection registers, but only the first error for a particular unit will 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.

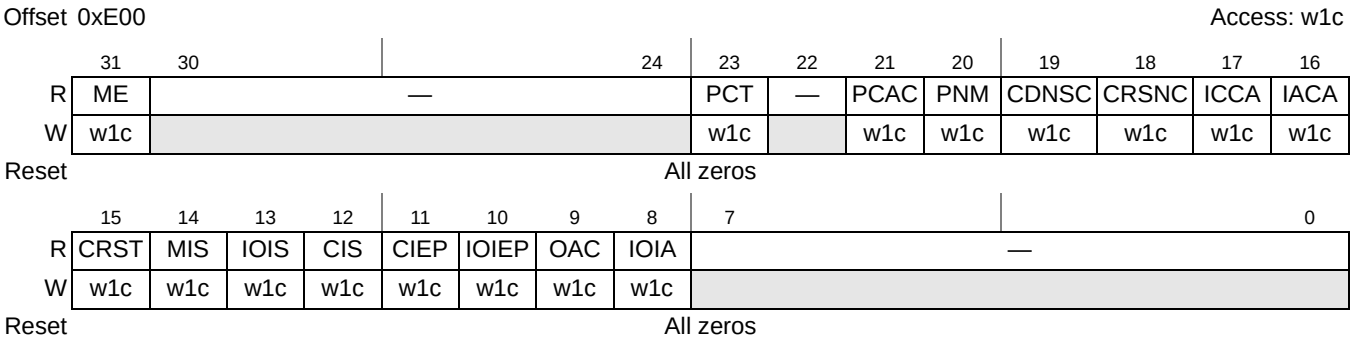


Figure 17-35. PCI Express Error Detect Register (PEX_ERR_DR)

Table 17-41 describes the fields of the PCI Express error detect register.

Table 17-41. PCI Express Error Detect Register Field Descriptions

Bits	Name	Description
31	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.
30–24	—	Reserved
23	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.
22	—	Reserved
21	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.

Table 17-41. PCI Express Error Detect Register Field Descriptions (Continued)

Bits	Name	Description
20	PNM	PCI Express no map. Detect an inbound transaction that was not mapped to any inbound windows. In RC mode, a completion without data (CPL) packet with a UR completion status is sent back 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 will not set this bit. 1 A no-map transaction was detected in RC mode. 0 No no-map transaction detected.
19	CDNSC	Completion with data not successful. A completion with data packet was received with a non successful status (i.e. UR, CA or CRS status). 1 Completion with data non successful packet was detected. 0 No completion with data non successful packet detected.
18	CRSNC	CRS non configuration. A completion was detected for a non configuration cycle and with CRS status. 1 CRS non configuration packet was detected. 0 No CRS non configuration packet detected.
17	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
16	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
15	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 17.4.1.1.4**, *PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR),* on page 17-30 for more information. 1 A CRS threshold condition was detected for an outbound configuration transaction 0 No CRS threshold condition detected
14	MIS	Message invalid size. An outbound message transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. See **Section 17.3.3.1**, *Outbound ATMU Message Generation,* on page 17-12 for more information. 1 An invalid size outbound message transaction was detected 0 No invalid size outbound message transaction detected
13	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
12	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
11	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

Table 17-41. PCI Express Error Detect Register Field Descriptions (Continued)

Bits	Name	Description
10	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
9	OAC	Outbound ATMU crossing. An outbound crossing ATMU transaction was detected. 1 An outbound transaction that hits in one window and crosses overing it was detected 0 No outbound ATMU crossing condition detected
8	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
7–0	—	Reserved

17.4.1.5.2 PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

The PCI Express error interrupt enable register, shown in **Figure 17-36**, allows interrupts to be generated when the corresponding PCI Express error detect register bits are set.

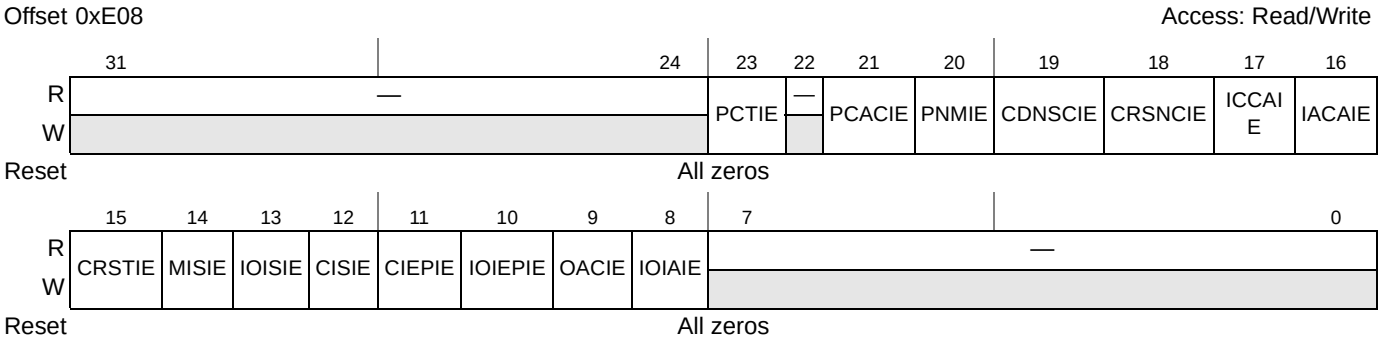


Figure 17-36. PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

Table 17-42 describes the fields of the PCI Express error interrupt enable register.

Table 17-42. PCI Express Error Interrupt Enable Register Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23	PCTIE	PCI Express completion time-out interrupt enable. When set and PEX_ERR_DR[PCT]=1 will generate an interrupt. 1 Enable PCI Express completion time-out interrupt generation 0 Disable PCI Express completion time-out interrupt generation
22	—	Reserved
21	PCACIE	PCI Express CA completion interrupt enable. When set and PEX_ERR_DR[PCAC]=1 will generate an interrupt. 1 Enable completion with CA status interrupt generation 0 Disable completion with CA status interrupt generation
20	PNMIE	PCI Express no map interrupt enable. When set and PEX_ERR_DR[PNM]=1 will generate an interrupt. 1 Enable no map PCI Express packet interrupt generation 0 Disable no map PCI Express packet interrupt generation

Table 17-42. PCI Express Error Interrupt Enable Register Field Descriptions (Continued)

Bits	Name	Description
19	CDNSCIE	Completion with data not successful interrupt enable. When this bit is set and PEX_ERR_DR[CDNSC] = 1 will generate an interrupt. 1 Enable completion with data non successful interrupt generation 0 Disable completion with data non successful interrupt generation
18	CRSNCIE	CRS non configuration interrupt enable. When this bit is set and PEX_ERR_DR[CRSNC] = 1 will generate an interrupt. 1 Enable CRS non configuration interrupt generation 0 Disable CRS non configuration interrupt generation
17	ICCAIE	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access interrupt enable. When set and PEX_ERR_DR[ICCA]=1 will generate 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.
16	IACAIE	Invalid ATMU configuration access. When set and PEX_ERR_DR[IACA]=1 will generate an interrupt. 1 Enable invalid ATMU configuration access interrupt generation 0 Disable invalid ATMU configuration access interrupt generation
15	CRSTIE	CRS thresholded interrupt enable. When set and PEX_ERR_DR[CRST]=1 will generate an interrupt. 1 Enable CRS threshold interrupt generation 0 Disable CRS threshold interrupt generation
14	MISIE	Message invalid size interrupt enable. When set and PEX_ERR_DR[MIS]=1 will generate an interrupt. 1 Enable invalid outbound message size interrupt generation 0 Disable invalid outbound message size interrupt generation
13	IOISIE	I/O invalid size interrupt enable. When set and PEX_ERR_DR[IOIS]=1 will generate an interrupt. 1 Enable invalid outbound I/O size interrupt generation 0 Disable invalid outbound I/O size interrupt generation
12	CISIE	Configuration invalid size interrupt enable. When set and PEX_ERR_DR[CIS]=1 will generate an interrupt. 1 Enable invalid outbound configuration size interrupt generation 0 Disable invalid outbound configuration size interrupt generation
11	CIEPIE	Configuration invalid EP interrupt enable. When set and PEX_ERR_DR[CIEP]=1 will generate an interrupt. 1 Enable outbound configuration transaction while in EP mode interrupt generation 0 Disable outbound configuration transaction in EP mode interrupt generation
10	IOIEPIE	I/O invalid EP interrupt enable. When set and PEX_ERR_DR[IOIEP]=1 will generate an interrupt. 1 Enable outbound I/O transaction EP mode interrupt generation 0 Disable outbound I/O transaction EP mode interrupt generation
9	OACIE	Outbound ATMU crossing interrupt enable. When set and PEX_ERR_DR[OAC]=1 will generate an interrupt. 1 Enable outbound crossing ATMU interrupt generation 0 Disable outbound crossing ATMU interrupt generation
8	IOIAIE	I/O address invalid enable. When set and PEX_ERR_DR[IOIA]=1 will generate an interrupt. 1 Enable greater than 4G I/O address interrupt generation 0 Disable greater than 4G I/O address interrupt generation
7–0	—	Reserved

17.4.1.5.3 PCI Express Error Disable Register (PEX_ERR_DISR)

The PCI Express error disable register, shown in **Figure 17-37**, controls the setting of the PCI Express error detect register's bits.

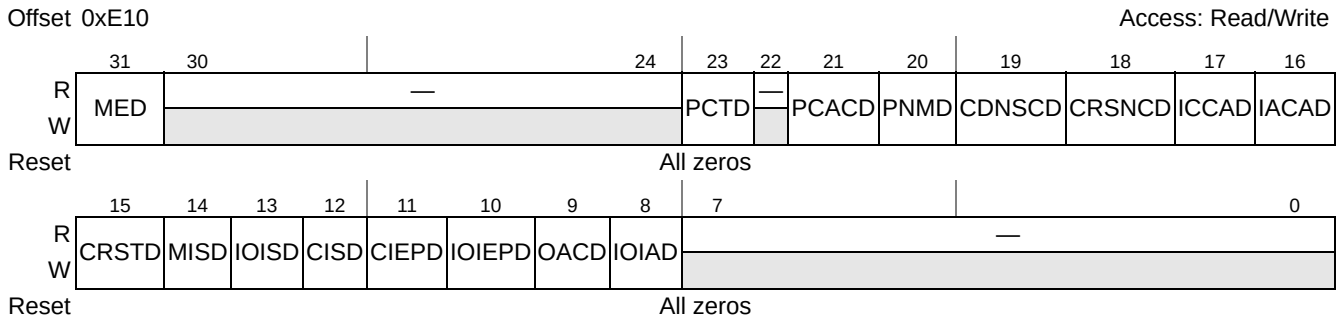


Figure 17-37. PCI Express Error Disable Register (PEX_ERR_DISR)

Table 17-43 describes the fields of the PCI Express error disable register.

Table 17-43. PCI Express Error Disable Register Field Descriptions

Bits	Name	Description
31	MED	Multiple errors disable. When set will disable the setting of PEX_ERR_DR[ME] bit. 1 Disable multiple errors detection 0 Enable multiple errors detection
30–24	—	Reserved
23	PCTD	PCI Express completion time-out disable. When set will disable the setting of PEX_ERR_DR[PET] bit. 1 Disable PCI Express completion time-out detection 0 Enable PCI Express completion time-out detection
22	—	Reserved
21	PCACD	PCI Express CA completion disable. When set will disable the setting of PEX_ERR_DR[PCAC] bit. 1 Disable completion with CA status detection 0 Enable completion with CA status detection
20	PNMD	PCI Express no map disable. When set will disable the setting of PEX_ERR_DR[PNM] bit. 1 Disable no map PCI Express packet detection 0 Enable no map PCI Express packet detection
19	CDNSCD	Completion with data not successful disable. When set will disable the setting of PEX_ERR_DR[CDNSC] bit. 1 Disable completion with data not successful detection 0 Enable completion with data not successful detection
18	CRSNCD	CRS non configuration disable. When set will disable the setting of PEX_ERR_DR[CRSNC] bit. 1 Disable CRS non configuration detection 0 Enable CRS non configuration detection
17	ICCAD	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access disable. When set will disable 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
16	IACAD	Invalid ATMU configuration access. When set will disable the setting of PEX_ERR_DR[IACA] bit. 1 Disable invalid ATMU configuration access detection 0 Enable invalid ATMU configuration access detection
15	CRSTD	CRS thresholded disable. When set will disable the setting of PEX_ERR_DR[CRST] bit. 1 Disable CRS threshold detection 0 Enable CRS threshold detection

Table 17-43. PCI Express Error Disable Register Field Descriptions (Continued)

Bits	Name	Description
14	MISD	Message invalid size disable. When set will disable the setting of PEX_ERR_DR[MIS] bit. 1 Disable invalid outbound message size detection 0 Enable invalid outbound message size detection
13	IOISD	I/O invalid size disable. When set will disable the setting of PEX_ERR_DR[IOIS] bit. 1 Disable invalid outbound I/O size detection 0 Enable invalid outbound I/O size detection
12	CISD	Configuration invalid size disable. When set will disable the setting of PEX_ERR_DR[CIS] bit. 1 Disable invalid outbound configuration size detection 0 Enable invalid outbound configuration size detection
11	CIEPD	Configuration invalid EP disable. When set will disable the setting of PEX_ERR_DR[CIEP] bit. 1 Disable outbound configuration transaction EP mode detection 0 Enable outbound configuration transaction EP mode detection
10	IOIEPD	I/O invalid EP disable. When set will disable 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
9	OACD	Outbound ATMU crossing disable. When set will disable the setting of PEX_ERR_DR[OAC] bit. 1 Disable outbound crossing ATMU detection 0 Enable outbound crossing ATMU detection
8	IOIAD	I/O invalid address disable. When set will disable 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
7–0	—	Reserved

17.4.1.5.4 PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

The PCI Express error capture status register, shown in **Figure 17-38**, 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.

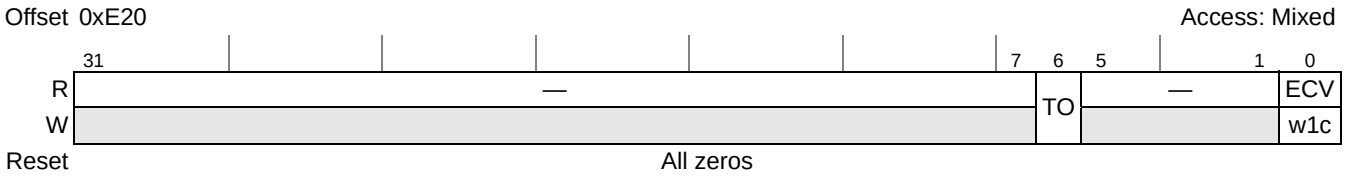


Figure 17-38. PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

Table 17-44 describes the fields of the PCI Express error capture status register.

Table 17-44. PCI Express Error Capture Status Register Field Descriptions

Bits	Name	Description
0–24	—	Reserved
6	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.
5–1	—	Reserved
0	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 will be done unless this bit is cleared by writing a 1 to it.

17.4.1.5.5 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.

PEX_ERR_CAP_R0 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in **Figure 17-39**.

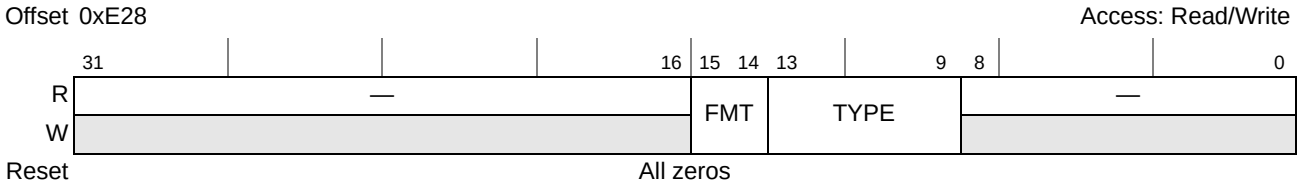


Figure 17-39. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source, Outbound Transaction

Table 17-45 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 17-45. PCI Express Error Capture Register 0 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
31–16	—	Reserved
15–14	FMT	PCI Express format. This field indicates the PCI Express packet format. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
13–9	TYPE	PCI Express type. This field indicates the PCI express packet type. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
8–0	—	Reserved

PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in **Figure 17-40**.

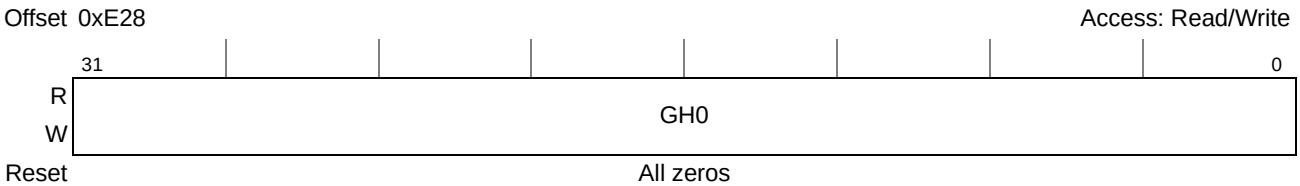


Figure 17-40. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) External Source, Inbound Transaction

Table 17-46 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 17-46. PCI Express Error Capture Register 0 Field Descriptions
External Source, Inbound Transaction

Bits	Name	Description
31–0	GH0	PCI Express 1st DW (4-byte) header. This field contains the PCI Express error packet's 1st DW (4-byte) header. 27–31 type 26–26 fmt 20–24 Rsv 17–19 TC 16 Rsv 14–15 length[9:8] 12–13 Rsv 10–11 Attr 9 EP 8 TD 0–7 length[7:0]

17.4.1.5.6 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.

PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in **Figure 17-41**.

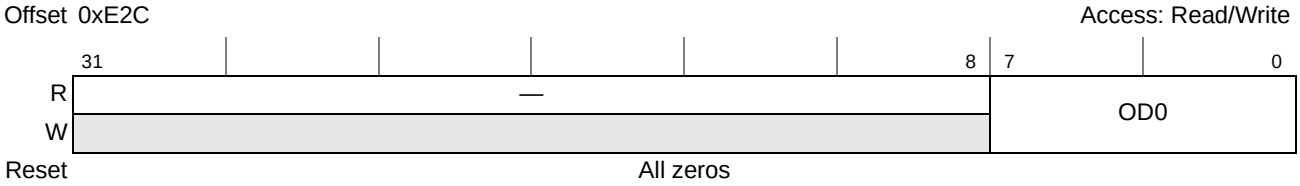


Figure 17-41. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)
Internal Source, Outbound Transaction

Table 17-47 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 17-47. PCI Express Error Capture Register 1 Field Descriptions
Internal Source, Outbound Transaction

Bits	Name	Description
31–8	—	Reserved
7–0	OD0	Internal platform transaction information. Reserved for factory debug.

PEX_ERR_CAP_R1 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in **Figure 17-42**.

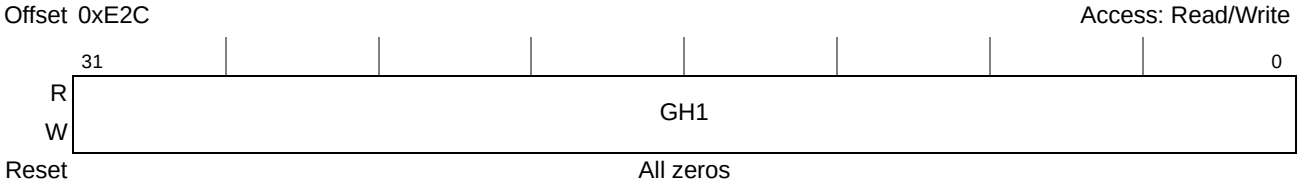


Figure 17-42. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)
External Source, Inbound Transaction

Table 17-48 describes the fields of PEX_ERR_CAP_R1 for the case when the error is caused by an inbound transaction from an external source.

Table 17-48. PCI Express Error Capture Register 1 Field Descriptions
External Source, Inbound Transaction

Bits	Name	Description
31–0	GH1	PEX 2nd DW (4-byte) header. This field contains the PCI Express error packet's 2nd DW (4-byte) header. 24–31 Comp ID[15:8] 16–23 Comp ID[7:0] 13–15 Comp Status 12 BCM 8–11 Byte Count[11:8] 0–7 Byte Count[7:0]

17.4.1.5.7 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.

PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in **Figure 17-43**.

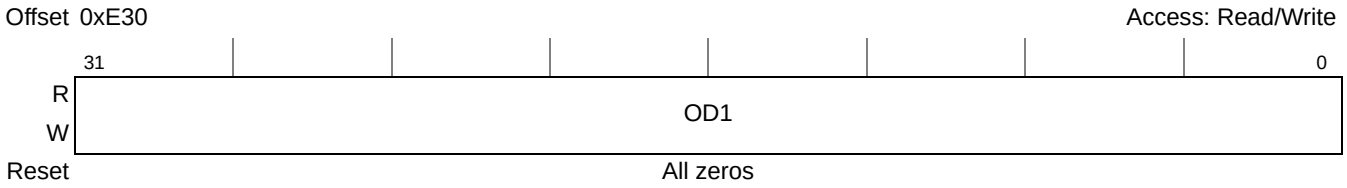


Figure 17-43. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) Internal Source, Outbound Transaction

Table 17-47 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 17-49. PCI Express Error Capture Register 2 Field Descriptions Internal Source, Outbound Transaction

Bit	Name	Description
31–0	OD1	Internal platform transaction information. Reserved for factory debug.

PEX_ERR_CAP_R2 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in **Figure 17-42**.

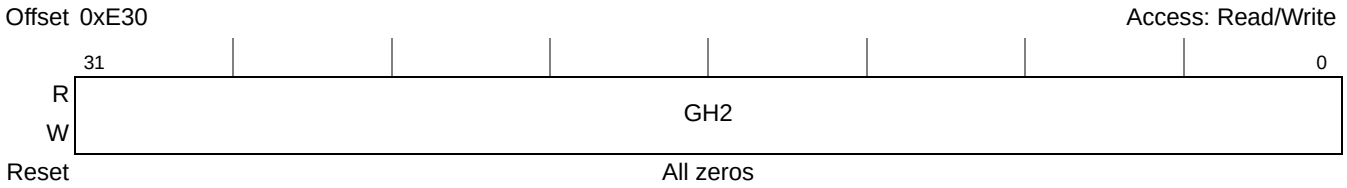


Figure 17-44. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) External Source, Inbound Transaction

Table 17-48 describes the fields of PEX_ERR_CAP_R2 for the case when the error is caused by an inbound transaction from an external source.

Table 17-50. PCI Express Error Capture Register 2 Field Descriptions
External Source, Inbound Transaction

Bits	Name	Description
31–0	GH2	PCI Express 3rd DW (4-byte) header. This field contains the PCI Express error packet's 3rd DW (4-byte) 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 Rsv

17.4.1.5.8 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.

PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in **Figure 17-45**.

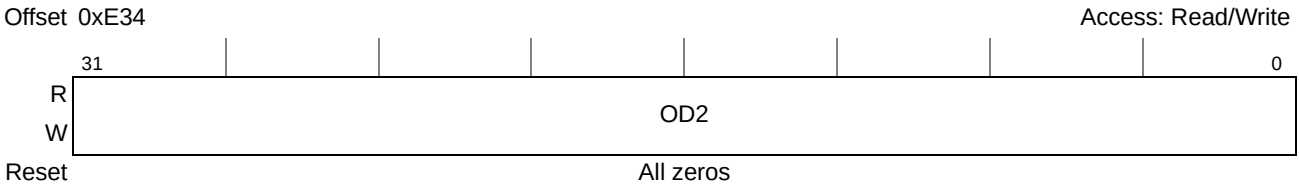


Figure 17-45. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)
Internal Source, Outbound Transaction

Table 17-47 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 17-51. PCI Express Error Capture Register 3 Field Descriptions
Internal Source, Outbound Transaction

Bits	Name	Description
31–0	OD2	Internal platform transaction information. Reserved for factory debug.

PEX_ERR_CAP_R3 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in **Figure 17-42**.

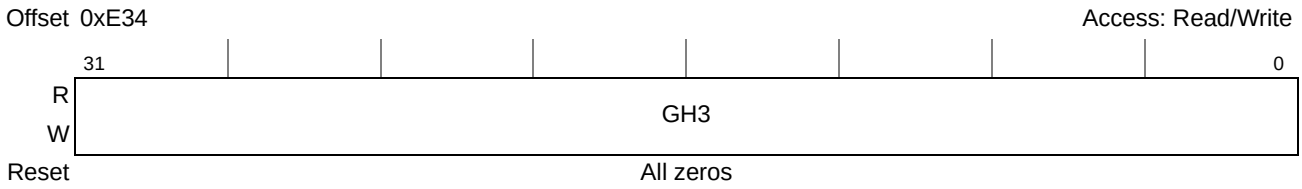


Figure 17-46. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)
External Source, Inbound Transaction

Table 17-48 describes the fields of PEX_ERR_CAP_R3 for the case when the error is caused by an inbound transaction from an external source.

Table 17-52. PEX Error Capture Register 3 Field Descriptions
External Source, Inbound Transaction

Bits	Name	Description
31–0	GH3	PEX 4th DW (4-byte) header. This field is a don't care.

17.4.1.6 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)

17.4.1.6.1 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 PCI Express bus, the device on that bus, the 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 17.3.2.2**, *Byte Order for Configuration Transactions*, on page 17-9 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_LTSSM_STAT) to check the status of link training before issuing external configuration requests.

17.4.1.6.2 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 bus number, and device number equal to the PCI Express controller's bus number and device number, and the function number is zero, then an internal PCI Express configuration cycle access is performed.
- If the 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 device number is 0, then a Type 0 configuration transaction is sent to the PCI Express link.
- If the 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 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.

17.4.1.6.3 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 bus number, device number, function number, register, and extended register number sent are decoded from the outbound translated PCI Express address.

- bus number[7:0] = PCI Express address[27:20]
- device number[4:0] = PCI Express address[19:15]
- function number[2:0] = PCI Express address[14:12]
- extended register number[3:0] = PCI Express address[11:8]
- register number[5:0] = PCI Express address[7:2]

A Type 0 configuration cycle is sent to the link if the bus number equals the secondary bus number (from the type 1 header) and device number is 0. A Type 1 configuration cycle is sent to the link if 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 will result in a response with error returned.

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.

17.4.1.6.4 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 up to 4096 bytes of the PCI Express configuration area. While in EP mode, the PCI Express controller does not support generating configuration accesses as a master. The PCI Express Controller Internal CSR registers are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers return all zeros.

All accesses to PEX_CONFIG_ADDR/PEX_CONFIG_DATA cause the device to access the internal configuration registers regardless of the bus number or 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 will get a response with an error.

17.4.1.7 PCI Compatible Configuration Headers

The first 64 bytes of the 256-byte PCI compatible configuration space consists of a predefined header that every PCI Express 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 17-47**.

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 17-47. 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.

17.4.1.7.1 Common PCI Compatible Configuration Header Registers

This section details the registers that are common to both type 0 and type 1 configuration headers.

17.4.1.7.2 PCI Express Vendor ID Register—Offset 0x00

The vendor ID register, shown in **Figure 17-48**, is used to identify the manufacturer of the device.

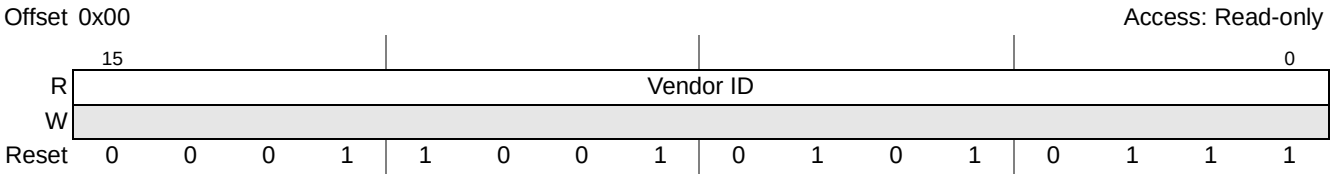


Figure 17-48. PCI Express Vendor ID Register

Table 17-53 describes the vendor ID register fields.

Table 17-53. PCI Express Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

17.4.1.7.3 PCI Express Device ID Register—Offset 0x02

The device ID register, shown in **Figure 17-49**, is used to identify the device.

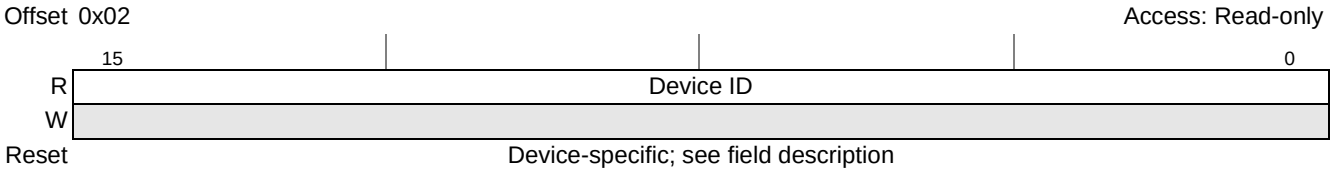


Figure 17-49. PCI Express Device ID Register

Table 17-54 describes the device ID register fields.

Table 17-54. PCI Express Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	Value is 0x181A

17.4.1.7.4 PCI Express Command Register—Offset 0x04

The command register, shown in **Figure 17-50**, provides control over the ability to generate and respond to PCI Express cycles.

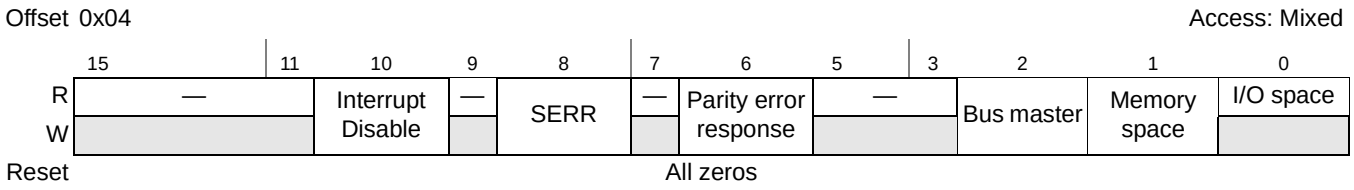


Figure 17-50. PCI Express Command Register

Table 17-55 describes the bits of the command register.

Table 17-55. 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 17.4.1.8.8**, *PCI Express Device Control Register—0x54,* on page 17-95 and the advance error reporting capability structure described in **Section 17.4.1.9.1** through **Section 17.4.1.9.12**.
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 17.4.1.8.8**, *PCI Express Device Control Register—0x54,* on page 17-95 and the advance error reporting capability structure described in **Section 17.4.1.9.1** through **Section 17.4.1.9.12**.
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.
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 will prevent 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 will prevent 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.

17.4.1.7.5 PCI Express Status Register—Offset 0x06

The status register, shown in **Figure 17-51**, 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							

	7	6	5	4	3	2		0
R	—	—	—	Capabilities list	Interrupt Status	—		
W								
Reset	0	0	0	1	0	0	0	0

Figure 17-51. PCI Express Status Register

The definition of each bit is given in **Table 17-56**.

Table 17-56. 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.
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 17.4.1.8.8**, *PCI Express Device Control Register—0x54,* on page 17-95 and the advance error reporting capability structure described in **Section 17.4.1.9.1** through **Section 17.4.1.9.12**.

17.4.1.7.6 PCI Express Revision ID Register—Offset 0x08

The revision ID register, shown in **Figure 17-52**, is used to identify the revision of the device.

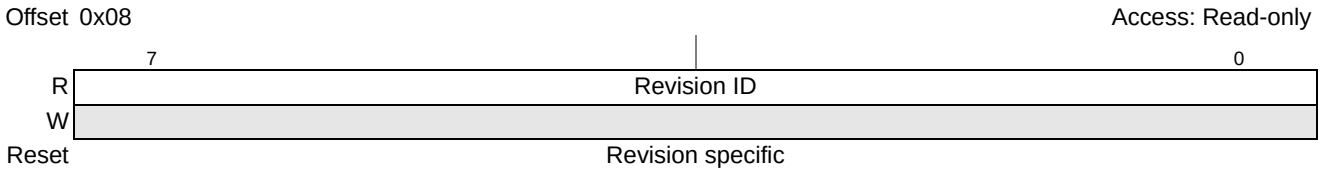


Figure 17-52. PCI Express Revision ID Register

Table 17-57 describes the revision ID register fields.

Table 17-57. PCI Express Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific.

17.4.1.7.7 PCI Express Class Code Register—Offset 0x09

The class code register, shown in **Figure 17-53**, 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.

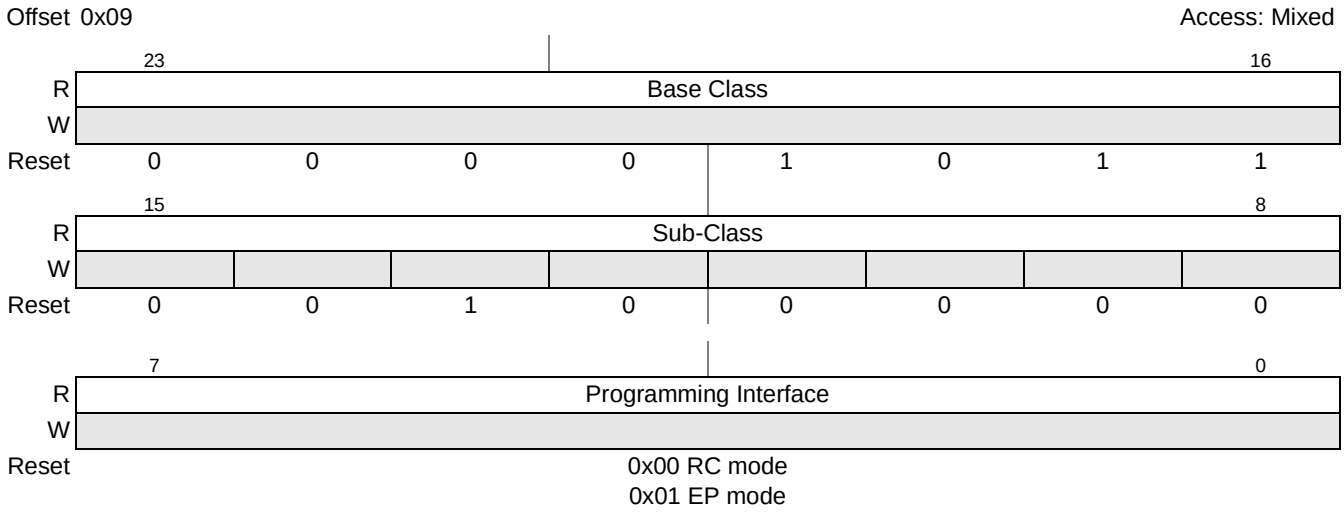


Figure 17-53. PCI Express Class Code Register

Table 17-58 describes the class code register fields.

Table 17-58. 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—RC mode 0x01—EP mode

17.4.1.7.8 PCI Express Cache Line Size Register—Offset 0x0C

The cache line size register, shown in **Figure 17-54**, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

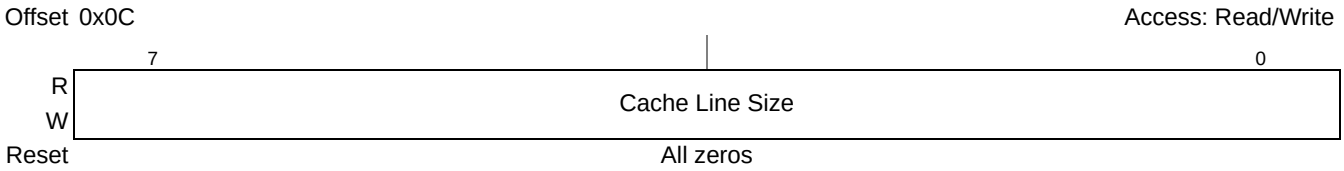


Figure 17-54. PCI Express Bus Cache Line Size Register

Table 17-59 describes the cache line size register.

Table 17-59. 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.

17.4.1.7.9 PCI Express Latency Timer Register—0x0D

The latency timer register, shown in **Figure 17-55**, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

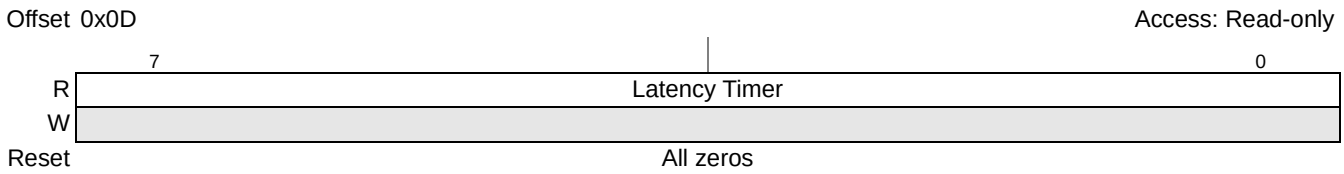


Figure 17-55. PCI Express Bus Latency Timer Register

Table 17-60 describes the PCI Express latency timer register (PLTR).

Table 17-60. 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.

17.4.1.7.10 PCI Express Header Type Register—0x0E

The PCI Express header type register, shown in **Figure 17-54**, is used to identify the layout of the PCI compatible header.

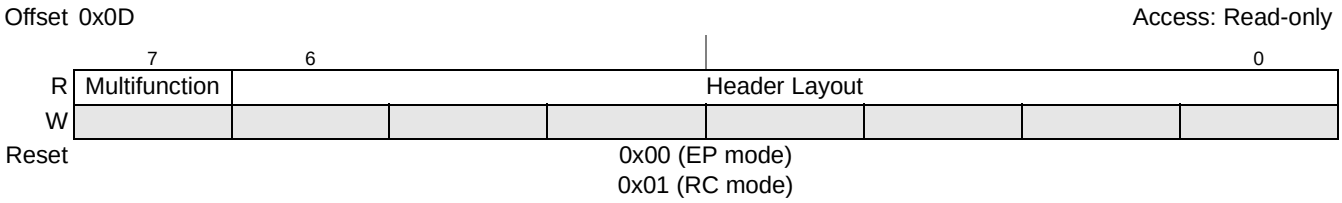


Figure 17-56. PCI Express Bus Latency Timer Register

Table 17-60 describes the PCI Express header type register.

Table 17-61. PCI Express Bus Latency Timer 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 17-57** for type 0 layout. 0x01 Root Complex. See **Figure 17-69** for type 1 layout. All other encodings reserved.

17.4.1.7.11 PCI Express BIST Register—0x0F

The BIST register is optional and reserved on the PCI Express controller.

17.4.1.7.12 Type 0 Configuration Header

The type 0 header is shown in **Figure 17-57**.

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 Registers			10 14 18 1C 20 24
			28
Subsystem ID		Subsystem Vendor ID	2C
			30
			Capabilities Pointer 34
Expansion ROM Base Address			38
MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line 3C

Figure 17-57. PCI Express PCI-Compatible Configuration Header—Type 0

Section 17.4.1.7.1, *Common PCI Compatible Configuration Header Registers*, on page 17-67 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.

17.4.1.7.13 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 17.4.1.4.6**, *PCI Express Inbound ATMU Registers*, on page 17-47 for more information. Base address register 0 at offset 0x10 is a special fixed 1-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 17-58**.

Offset 0x10 Access: Mixed

	31 … 20	19 … 4	3	2 1	0
R	ADDRESS	—	PREF	TYPE	MemSp
W					
Reset	All zeros				

Figure 17-58. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 17-62 describes the PCI Express configuration and status register base address register.

Table 17-62. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	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 17-59**.

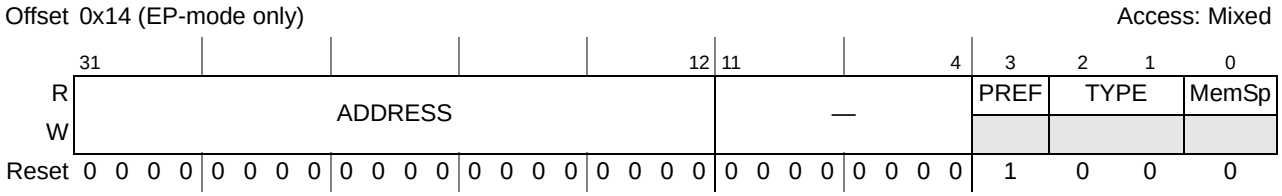


Figure 17-59. 32-Bit Memory Base Address Register (BAR1)

Table 17-63 describes the PCI Express 32-bit memory BAR fields.

Table 17-63. 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 17-60**.

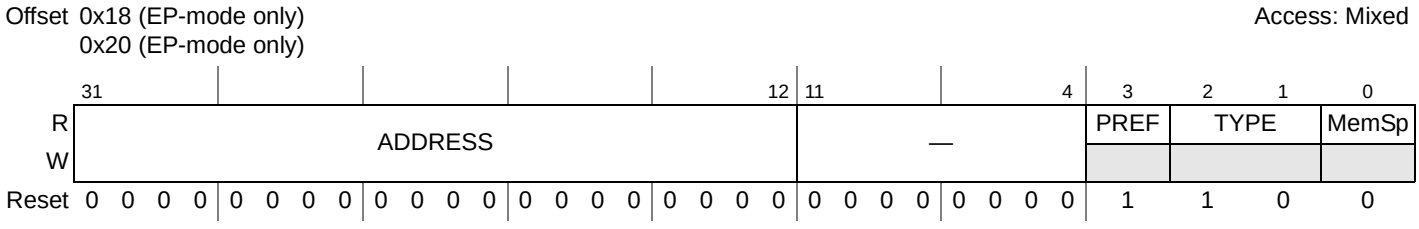


Figure 17-60. 64-Bit Low Memory Base Address Register

Table 17-64 describes the PCI Express 64-bit low memory BAR fields.

Table 17-64. 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].
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 17-61**.

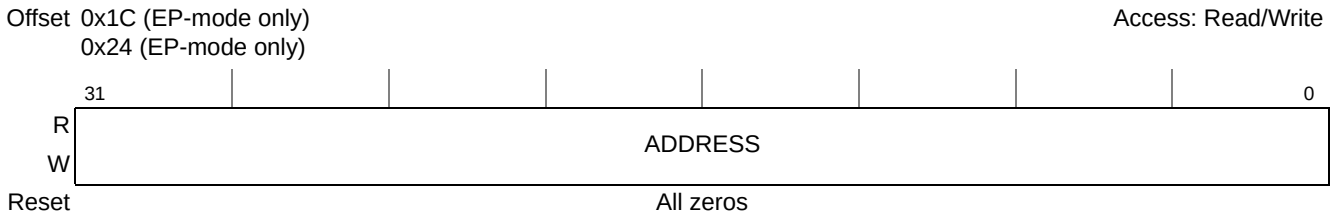


Figure 17-61. 64-Bit High Memory Base Address Register

Table 17-65 describes the PCI Express 64-bit low memory BAR fields.

Table 17-65. 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.

17.4.1.7.14 PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C

The PCI Express subsystem vendor ID register is used to identify the subsystem.

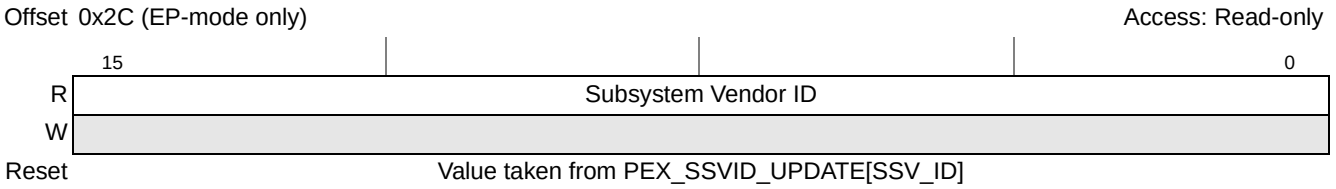


Figure 17-62. PCI Express Subsystem Vendor ID Register

Table 17-66. PCI Express Subsystem Vendor ID Register Field Description

Bits	Name	Description
15–0	Subsystem Vendor ID	The value for subsystem vendor ID is determined by the PCI Express subsystem vendor ID update register. See **Section 17.4.1.9.19**, *PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,* on page 17-119 for more information.

17.4.1.7.15 PCI Express Subsystem ID Register (EP-Mode Only)—0x2E

The PCI Express subsystem ID register is used to identify the subsystem.

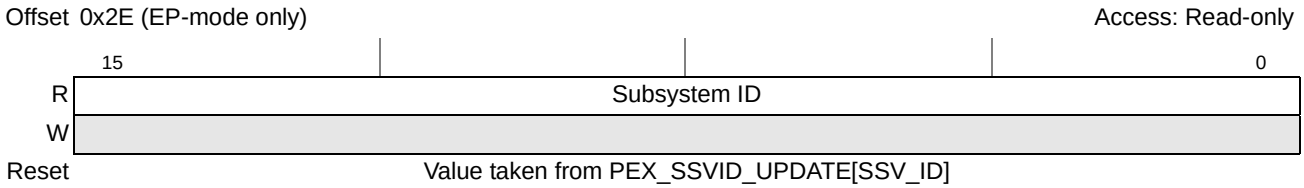


Figure 17-63. PCI Express Subsystem ID Register

Table 17-67. PCI Express Subsystem ID Register Field Description

Bits	Name	Description
15–0	Subsystem ID	The value for subsystem ID is determined by the PCI Express subsystem vendor ID update register. See **Section 17.4.1.9.19**, *PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,* on page 17-119 for more information.

17.4.1.7.16 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

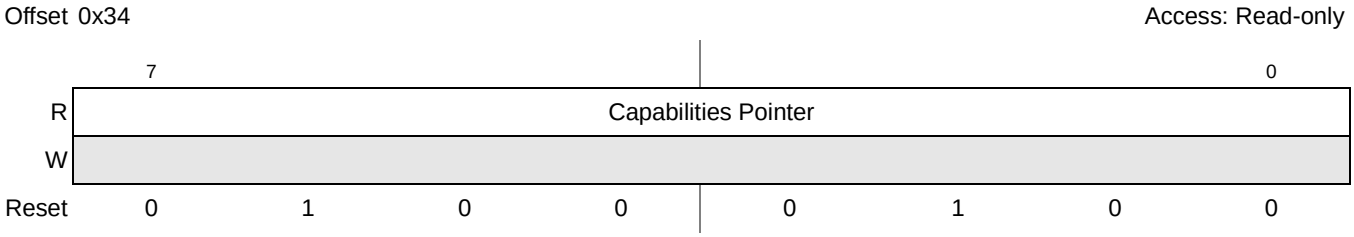


Figure 17-64. Capabilities Pointer Register

Table 17-68. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to **Section 17.4.1.8**, *PCI Compatible Device-Specific Configuration Space,* on page 17-90 for more information.

17.4.1.7.17 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.

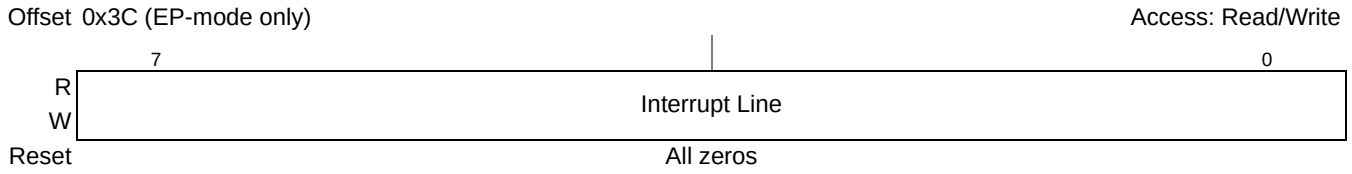


Figure 17-65. PCI Express Interrupt Line Register

Table 17-69. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

17.4.1.7.18 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

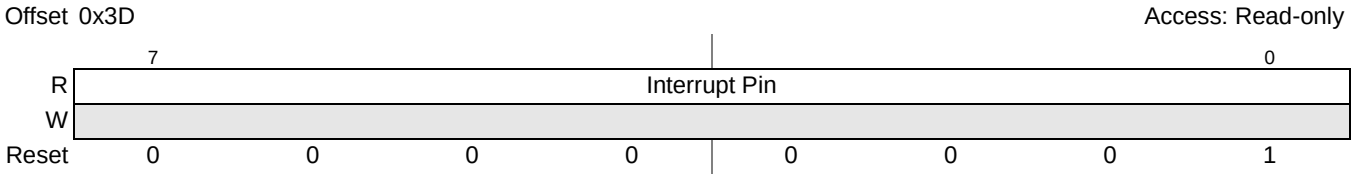


Figure 17-66. PCI Express Interrupt Pin Register

Table 17-70. 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.

17.4.1.7.19 PCI Express Minimum Grant Register (EP-Mode Only)—0x3E

This register does not apply to PCI Express. It is present for legacy purposes.

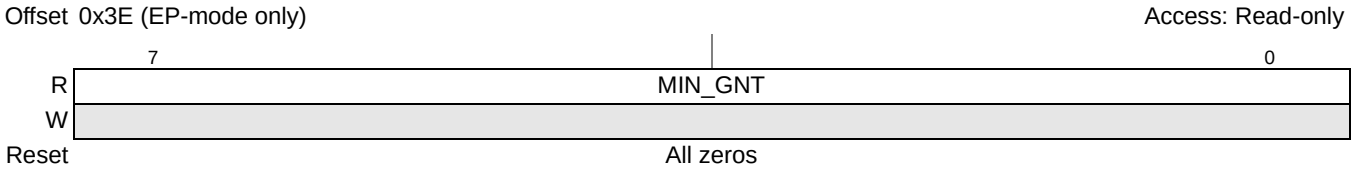


Figure 17-67. PCI Express Maximum Grant Register (MAX_GNT)

Table 17-71. PCI Express Maximum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	Does not apply for PCI Express.

17.4.1.7.20 PCI Express Maximum Latency Register (EP-Mode Only)—0x3F

This register does not apply to PCI Express. It is present for legacy purposes.

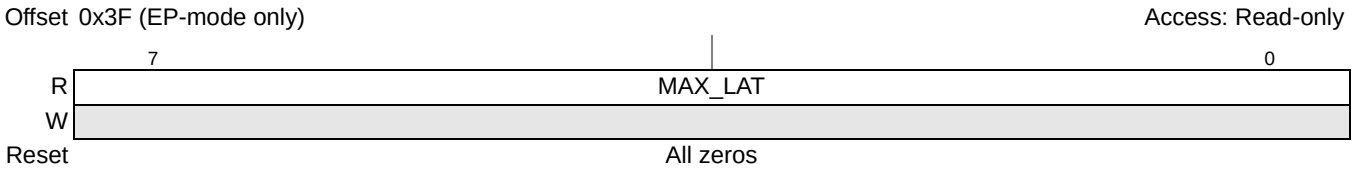


Figure 17-68. PCI Express Maximum Latency Register (MAX_LAT)

Table 17-72. PCI Express Maximum Latency Register Field Description

Bits	Name	Description
7–0	MAX_LAT	Does not apply for PCI Express.

17.4.1.7.21 Type 1 Configuration Header

The type 1 header is shown in **Figure 17-69**.

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
Bridge Control		Interrupt Pin	Interrupt Line	3C

Figure 17-69. PCI Express PCI-Compatible Configuration Header—Type 1

Section 17.4.1.7.1, *Common PCI Compatible Configuration Header Registers*, on page 17-67 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.

17.4.1.7.22 PCI Express Base Address Register 0—0x10

Base address register 0 at offset 0x10 is a special fixed 1-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 17-58**.

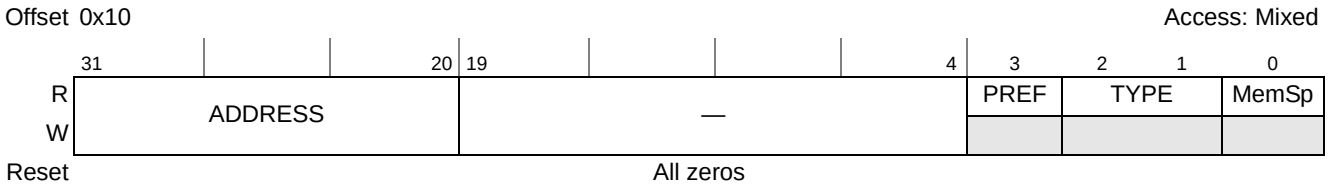


Figure 17-70. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 17-62 describes the PCI Express configuration and status register base address register.

Table 17-73. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

17.4.1.7.23 PCI Express Primary Bus Number Register—Offset 0x18

The primary bus number register is shown in **Figure 17-71**.

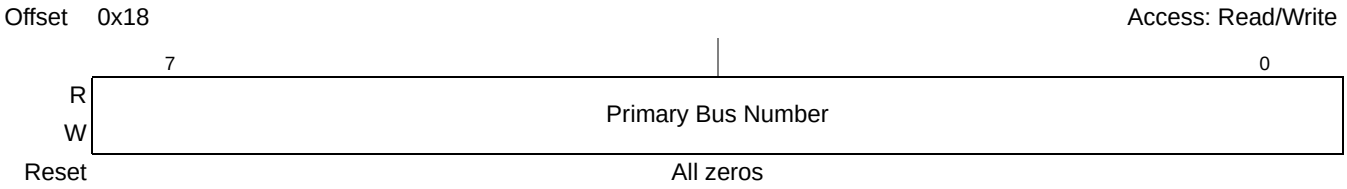


Figure 17-71. PCI Express Primary Bus Number Register

Table 17-74 describes the primary bus number register fields.

Table 17-74. 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.

17.4.1.7.24 PCI Express Secondary Bus Number Register—Offset 0x19

The secondary bus number register is shown in **Figure 17-72**.

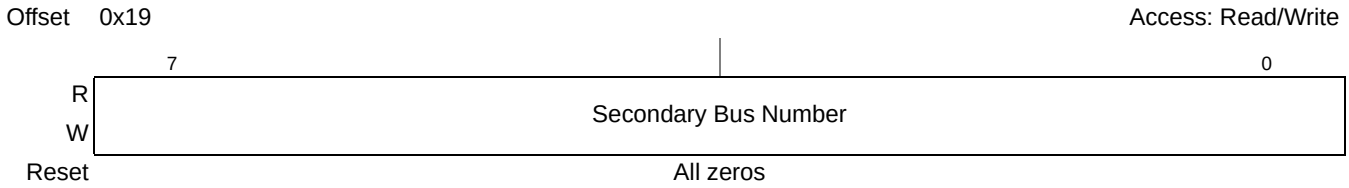


Figure 17-72. PCI Express Secondary Bus Number Register

Table 17-75 describes the secondary bus number register fields.

Table 17-75. 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.

17.4.1.7.25 PCI Express Subordinate Bus Number Register—Offset 0x1A

The subordinate bus number register is shown in **Figure 17-73**.

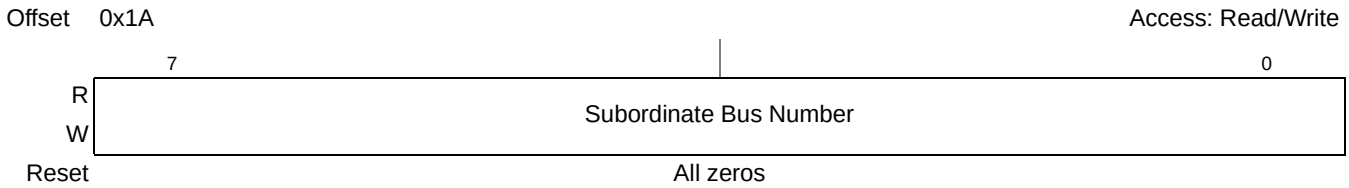


Figure 17-73. PCI Express Subordinate Bus Number Register

Table 17-76 describes the subordinate bus number register fields.

Table 17-76. 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.

17.4.1.7.26 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.

17.4.1.7.27 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 17-73**.

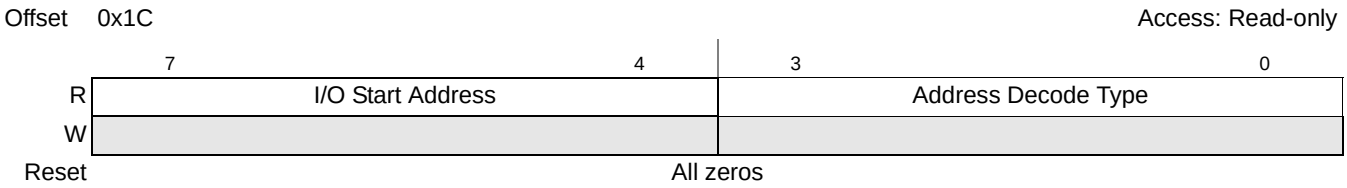


Figure 17-74. PCI Express I/O Base Register

Table 17-76 describes the I/O base register fields.

Table 17-77. 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.

17.4.1.7.28 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 17-73**.

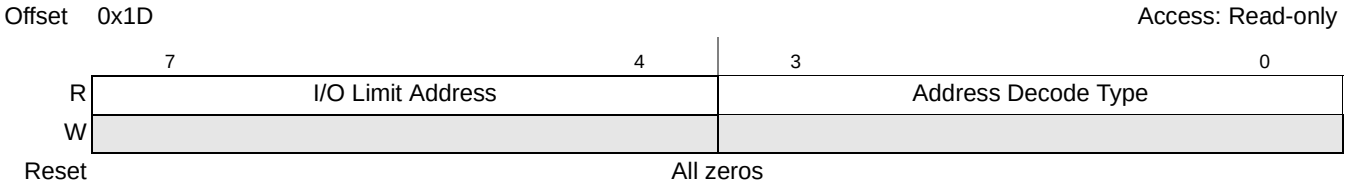


Figure 17-75. PCI Express I/O Limit Register

Table 17-76 describes the I/O limit register fields.

Table 17-78. 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.

17.4.1.7.29 PCI Express Secondary Status Register—0x1E

The PCI Express secondary status register is shown in **Figure 17-76**. 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. See **Section 17.4.1.9.22**, *Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0*, on page 17-121 for more information.

Offset 0x1E — Access: Mixed

	15	14	13	12	11	10	9	8	7	0
R	DPE	SSE	RMA	RTA	STA	—		MDPE	—	
W	w1c	w1c	w1c	w1c	w1c			w1c		
Reset	All zeros									

Figure 17-76. PCI Express Secondary Status Register

Table 17-79 describes the PCI Express secondary status register fields.

Table 17-79. 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

17.4.1.7.30 PCI Express Memory Base Register—0x20

The memory base register is shown in **Figure 17-77**.

Figure 17-77. PCI Express Memory Base Register

Table 17-80 describes the memory base register fields.

Table 17-80. 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

17.4.1.7.31 PCI Express Memory Limit Register—0x22

The memory limit register is shown in **Figure 17-78**.

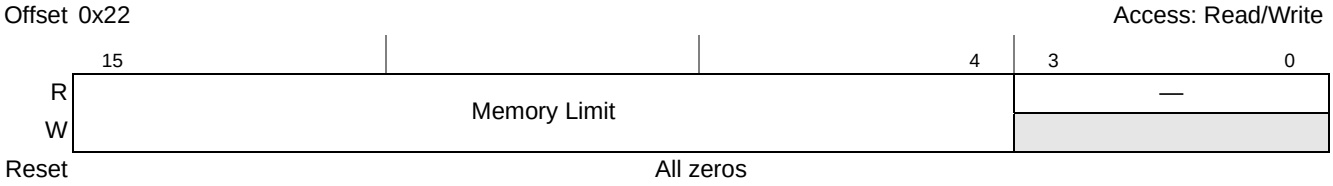


Figure 17-78. PCI Express Memory Limit Register

Table 17-81 describes the memory base register fields.

Table 17-81. 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 will result in unsupported request response.
3–0	—	Reserved

17.4.1.7.32 PCI Express Prefetchable Memory Base Register—0x24

The prefetchable memory base register is shown in **Figure 17-79**.

Figure 17-79. PCI Express Prefetchable Memory Base Register

Table 17-82 describes the prefetchable memory base register fields.

Table 17-82. 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.

17.4.1.7.33 PCI Express Prefetchable Memory Limit Register—0x26

The prefetchable memory limit register is shown in **Figure 17-80**.

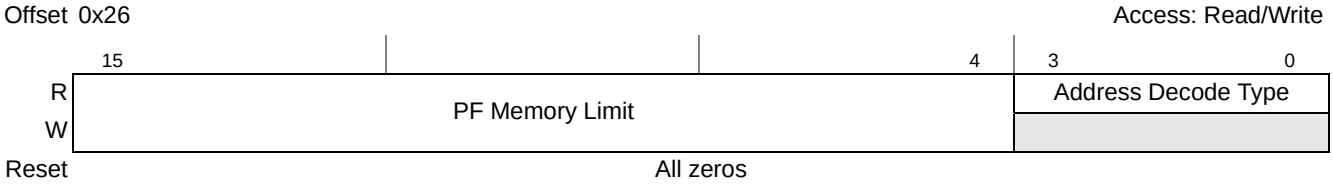


Figure 17-80. PCI Express Prefetchable Memory Limit Register

Table 17-83 describes the prefetchable memory limit register fields.

Table 17-83. 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.

17.4.1.7.34 PCI Express Prefetchable Base Upper 32 Bits Register—0x28

The PCI Express prefetchable memory base upper 32 bits register is shown in **Figure 17-81**.

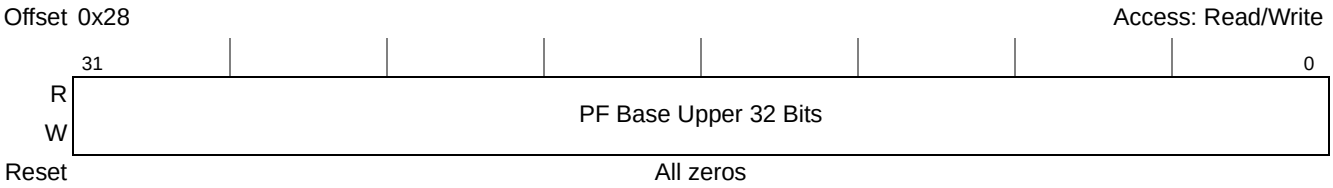


Figure 17-81. PCI Express Prefetchable Base Upper 32 Bits Register

Table 17-84 describes the PCI Express prefetchable memory base upper 32 bits register fields.

Table 17-84. 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.

17.4.1.7.35 PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C

The PCI Express prefetchable memory base upper 32 bits register is shown in **Figure 17-82**.

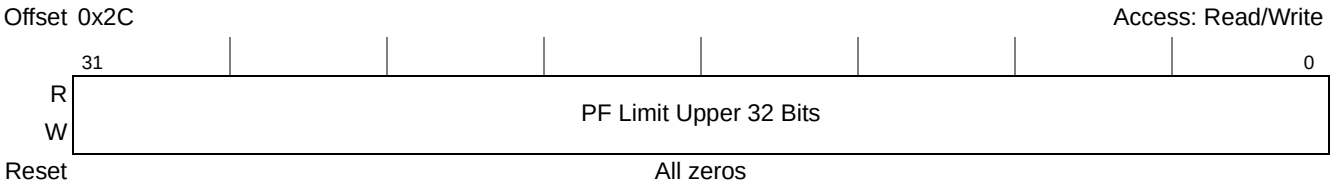


Figure 17-82. PCI Express Prefetchable Limit Upper 32 Bits Register

Table 17-85 describes the PCI Express prefetchable memory limit upper 32 bits register fields.

Table 17-85. 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.

17.4.1.7.36 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 17-83**.

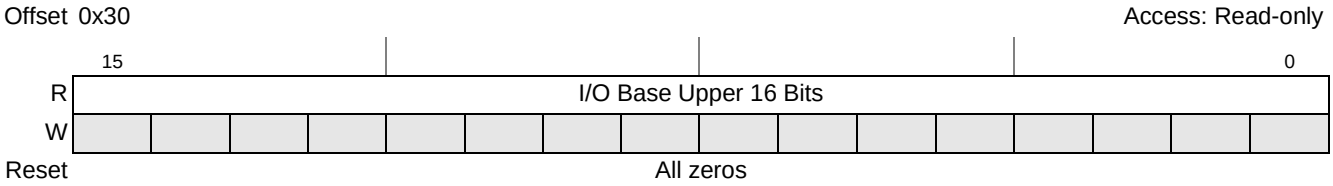


Figure 17-83. PCI Express I/O Base Upper 16 Bits Register

Table 17-86 describes the I/O base upper 16 bits register fields.

Table 17-86. 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.

17.4.1.7.37 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 17-84**.

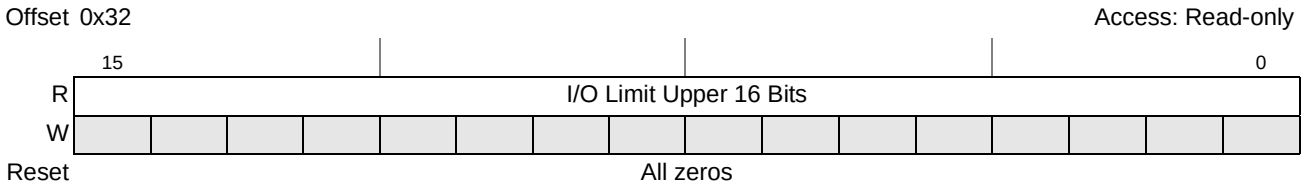


Figure 17-84. PCI Express I/O Limit Upper 16 Bits Register

Table 17-87 describes the I/O limit upper 16 bits register fields.

Table 17-87. 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.

17.4.1.7.38 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

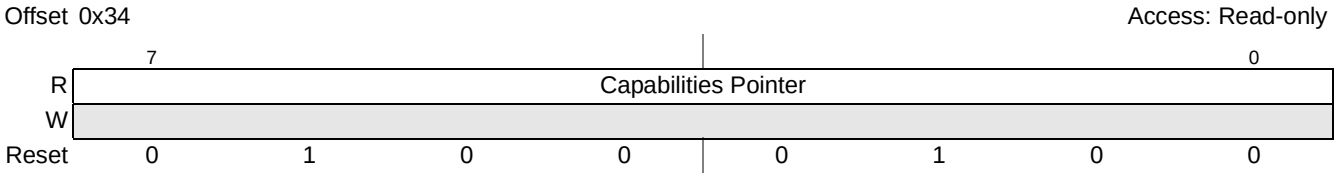


Figure 17-85. Capabilities Pointer Register

Table 17-88. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to **Section 17.4.1.8**, *PCI Compatible Device-Specific Configuration Space,* on page 17-90," for more information.

17.4.1.7.39 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.

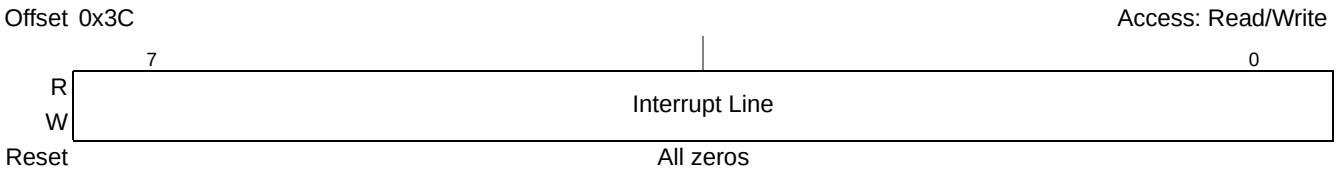


Figure 17-86. PCI Express Interrupt Line Register

Table 17-89. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

17.4.1.7.40 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

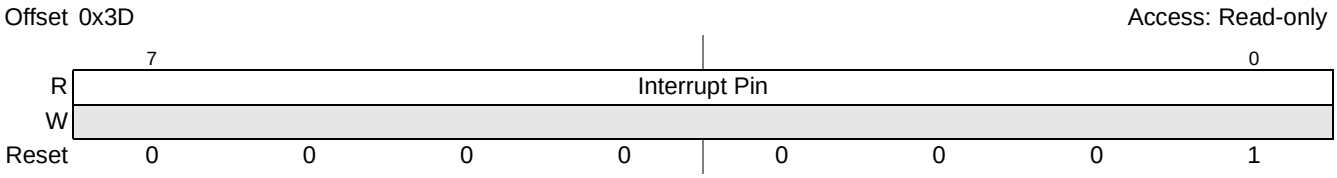


Figure 17-87. PCI Express Interrupt Pin Register

Table 17-90. 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.

17.4.1.7.41 PCI Express Bridge Control Register—0x3E

The PCI Express bridge control register is shown in **Figure 17-88**.

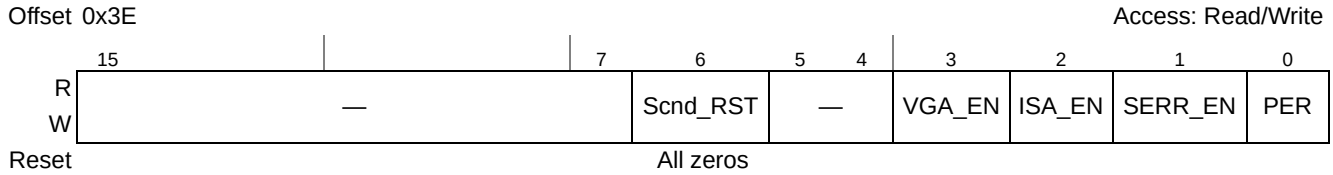


Figure 17-88. PCI Express Bridge Control Register

Table 17-91 describes the PCI Express bridge control register fields.

Table 17-91. 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.

17.4.1.8 PCI Compatible Device-Specific Configuration Space

The PCI compatible device-specific configuration space is a PCI compatible configuration space from 0x40 to 0xFF (just above the 64-byte PCI-compatible configuration header).

Reserved

			Address Offset (Hex)
PCI-Compatible Configuration Header (See **Section 17.4.1.7**, *PCI Compatible Configuration Headers* for more information.)			00 3F
			40
Power Mgmt Capabilities	Next Pointer (0x4C)	Power Mgmt Capability ID	44
Data	Power Management Status & Control		48
PCI Express Capabilities	Next Pointer (0x70 — EP mode) (NULL — RC mode)	PCI Express Capability ID	4C
Device Capabilities			50
Device Status	Device Control		54
Link Capabilities			58
Link Status	Link Control		5C
Slot Capabilities			60
Slot Status	Slot Control		64
	Root Control (RC mode only)		68
Root Status			6C
MSI Message Control	Next Pointer (NULL)	MSI Message Capability ID	70
MSI Message Address			74
MSI Upper Message Address			78
	MSI Message Data		7C
			80 FF

Figure 17-89. PCI Compatible Device-Specific Configuration Space

17.4.1.8.1 PCI Express Power Management Capability ID Register—0x44

The PCI Express power management capability ID register is shown in **Figure 17-90**.

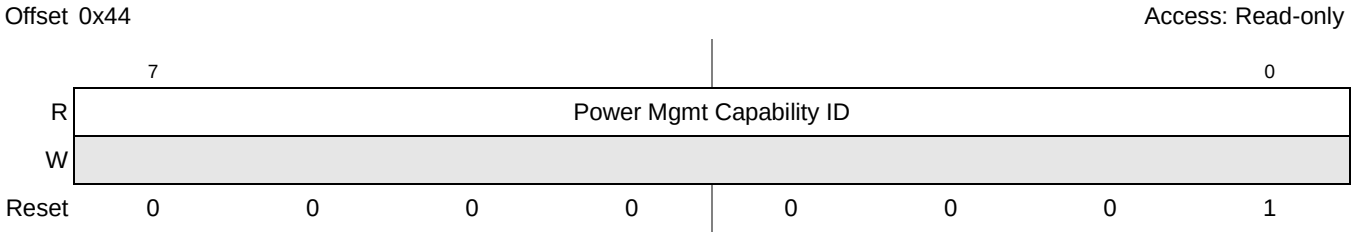


Figure 17-90. PCI Express Power Management Capability ID Register

Table 17-92. PCI Express Power Management Capability ID Register Field Description

Bits	Name	Description
7–0	Power Mgmt Capability ID	Power Management = 0x01

17.4.1.8.2 PCI Express Power Management Capabilities Register—0x46

The PCI Express power management capabilities register is shown in **Figure 17-91**.

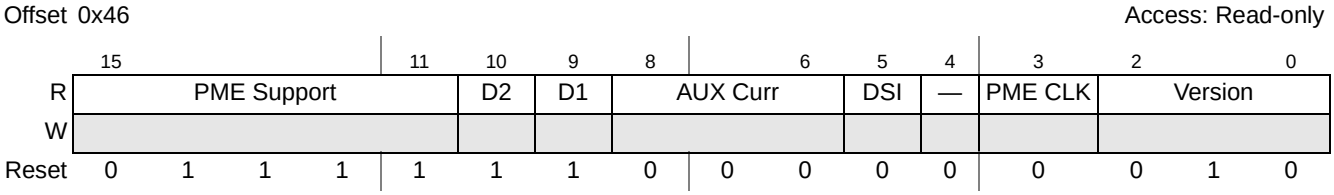


Figure 17-91. PCI Express Power Management Capabilities Register

Table 17-93. 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	Version 1.0a

17.4.1.8.3 PCI Express Power Management Status and Control Register—0x48

The PCI Express power management status and control register is shown in **Figure 17-92**.

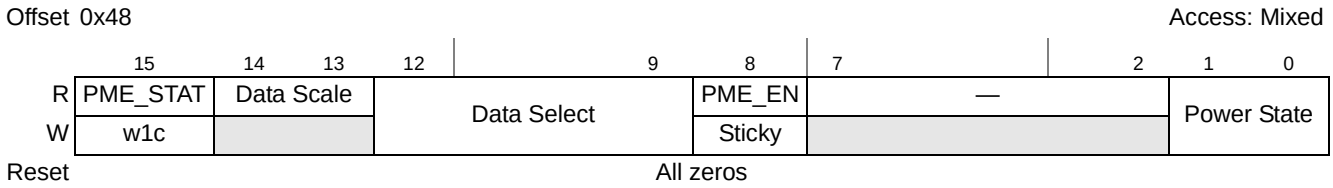


Figure 17-92. PCI Express Power Management Status and Control Register

Table 17-94. PCI Express Status and Control Register Field Description

Bits	Name	Description
15	PME_STAT	PME Status
14–13	Data Scale	Obtained directly from *PCI Express™ Base Specification, Revision 1.0a*
12–9	Data Select	Obtained directly from *PCI Express™ Base Specification, Revision 1.0a*
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

17.4.1.8.4 PCI Express Power Management Data Register—0x4B

The PCI Express power management data register is shown in **Figure 17-93**.

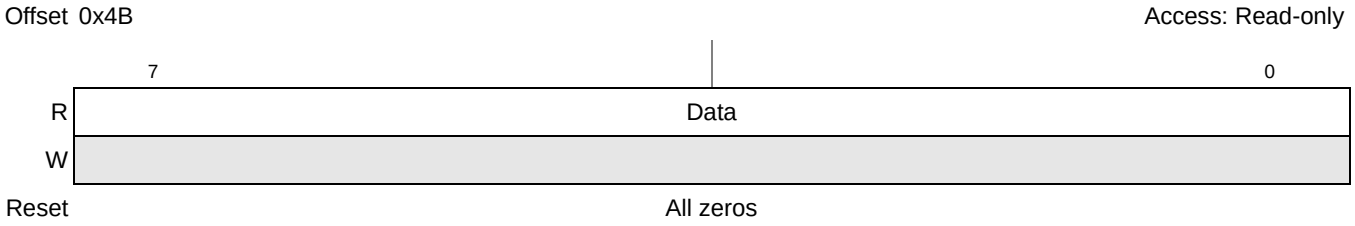


Figure 17-93. PCI Express Power Management Data Register

Table 17-95. PCI Express Power Management Data Register Field Description

Bits	Name	Description
7–0	Data	Obtained from *PCI Express™ Base Specification, Revision 1.0a*

17.4.1.8.5 PCI Express Capability ID Register—0x4C

The PCI Express capability ID register is shown in **Figure 17-94**.

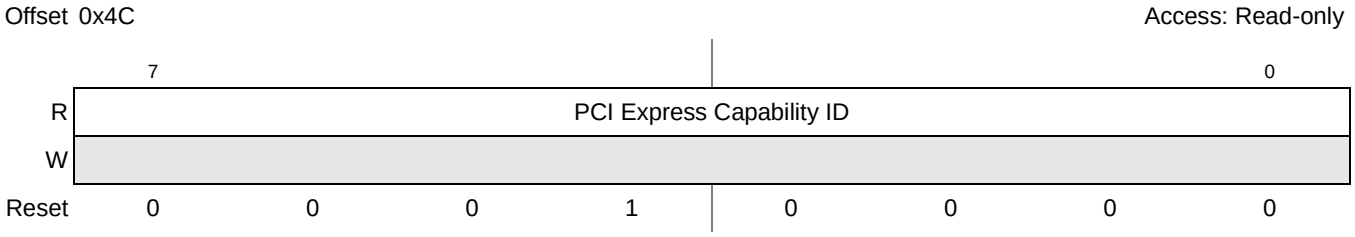


Figure 17-94. PCI Express Capability ID Register

Table 17-96. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	PCI Express Capability ID	PCI Express = 0x10

17.4.1.8.6 PCI Express Capabilities Register—0x4E

The PCI Express capabilities register is shown in **Figure 17-95**.

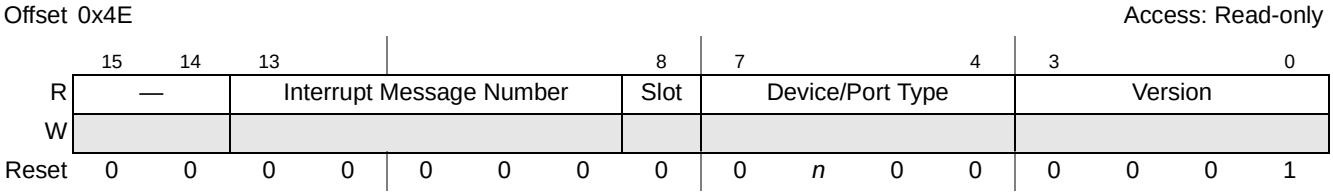


Figure 17-95. PCI Express Capabilities Register

Table 17-97. 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. Must be 1h for this specification.

17.4.1.8.7 PCI Express Device Capabilities Register—0x50

The PCI Express device capabilities register is shown in **Figure 17-96**.

Offset 0x50 Access: Read-only

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

	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

Figure 17-96. PCI Express Device Capabilities Register

Table 17-98. PCI Express Device Capabilities Register Field Description

Bits	Name	Description
31–28	—	Reserved
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

17.4.1.8.8 PCI Express Device Control Register—0x54

The PCI Express device control register is shown in **Figure 17-97**.

Offset 0x54 Access: Read/write

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

Figure 17-97. PCI Express Device Control Register

Table 17-99. PCI Express Device Control Register Field Description

Bits	Name	Description
15	—	Reserved
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

17.4.1.8.9 PCI Express Device Status Register—0x56

The PCI Express device status register is shown in **Figure 17-98**.

Offset 0x56 Access: Mixed

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—										TP	APD	URD	FED	NFED	CED
W													w1c	w1c	w1c	w1c
Reset	All zeros															

Figure 17-98. PCI Express Device Status Register

Table 17-100. PCI Express Device Status Register Field Description

Bits	Name	Description
15–6	—	Reserved
5	TP	Transactions pending
4	APD	AUX power detected
3	URD	Unsupported request detected
2	FED	Fatal error detected
1	NFED	Non-fatal error detected
0	CED	Correctable error detected

17.4.1.8.10 PCI Express Link Capabilities Register—0x58

The PCI Express link capabilities register is shown in **Figure 17-99**.

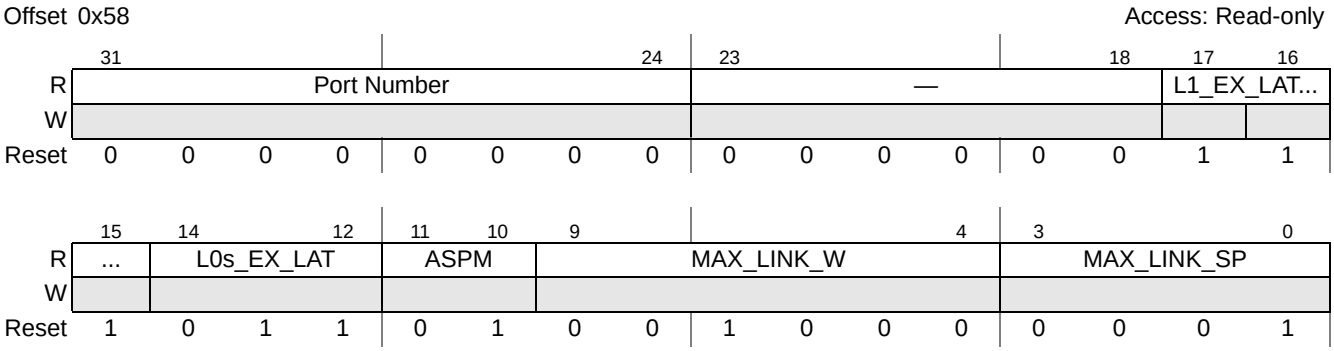


Figure 17-99. PCI Express Link Capabilities Register

Table 17-101. PCI Express Link Capabilities Register Field Description

Bits	Name	Description
31–24	Port Number	
23–18	—	Reserved
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

17.4.1.8.11 PCI Express Link Control Register—0x5C

The PCI Express link control register is shown in **Figure 17-100**.

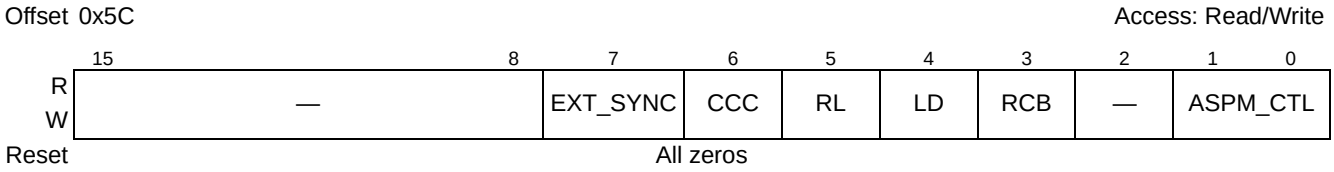


Figure 17-100. PCI Express Link Control Register

Table 17-102. PCI Express Link Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
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
3	RCB	Read completion boundary
2	—	Reserved
1–0	ASPM_CTL	Active state power management (ASPM) control

17.4.1.8.12 PCI Express Link Status Register—0x5E

The PCI Express link status register is shown in **Figure 17-101**.

Offset 0x5E Access: Read-only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—			SCC	LT	—	NEG_LINK_W						LINK_SP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 17-101. PCI Express Link Status Register

Table 17-103. PCI Express Link Status Register Field Description

Bits	Name	Description
15–13	—	Reserved
12	SCC	Slot clock configuration
11	LT	Link training
10	—	Reserved.
9–4	NEG_LINK_W	Negotiated link width
3–0	LINK_SP	Link speed

17.4.1.8.13 PCI Express Slot Capabilities Register—0x60

The PCI Express slot capabilities register is shown in **Figure 17-102**.

Offset 0x60 Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Physical Slot Number													—		SPLS
W																
Reset	All zeros															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SPLS	SPLV								HPC	HPS	PIP	AIP	MRLSP	PCP	ABP
W																
Reset	All zeros															

Figure 17-102. PCI Express Slot Capabilities Register

Table 17-104. 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 within the chassis. These registers should be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18–17	—	Reserved
16–15	SPLS	Slot power limit scale.
14–7	SPLV	Slot power limit value.
6	HPD	Hot plug capable.
5	HPS	Hot plug surprise.
4	PIP	Power indicator present.

Table 17-104. PCI Express Slot Capabilities Register Field Description (Continued)

Bits	Name	Description
3	AIP	Attention indicator present.
2	MRLSP	MRL sensor present.
1	PCP	Power controller present.
0	ABP	Attention button present.

17.4.1.8.14 PCI Express Slot Control Register—0x64

The PCI Express slot control register is shown in **Figure 17-103**.

Offset 0x64 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	All zeros									

Figure 17-103. PCI Express Slot Control Register

Table 17-105. PCI Express Slot Control Register Field Description

Bits	Name	Description
15–11	—	Reserved
10	PCC	Power controller control.
9–8	PIC	Power indicator control.
7–6	AIC	Attention indicator control.
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.

17.4.1.8.15 PCI Express Slot Status Register—0x66

The PCI Express slot status register is shown in **Figure 17-104**.

Offset 0x66 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	0	0	0	0

Figure 17-104. PCI Express Slot Status Register

Table 17-106. PCI Express Slot Status Register Field Descriptions

Bits	Name	Description
15–7	—	Reserved
6	PDS	Presence detect state. This bit indicates the presence of a card in the slot. 0 Slot empty 1 Card is present
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.

17.4.1.8.16 PCI Express Root Control Register (RC Mode Only)—0x68

The PCI Express root control register is shown in **Figure 17-105**.

Offset 0x68 Access: Read/Write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—												PMEIE	SEFEE	SENFEE	SECEE
Reset	All zeros															

Figure 17-105. PCI Express Root Control Register

Table 17-107. 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.
1	SENFEE	System error on non-fatal error enable.
0	SECEE	System error on correctable error enable.

17.4.1.8.17 PCI Express Root Status Register (RC Mode Only)—0x6C

The PCI Express root status register is shown in **Figure 17-106**.

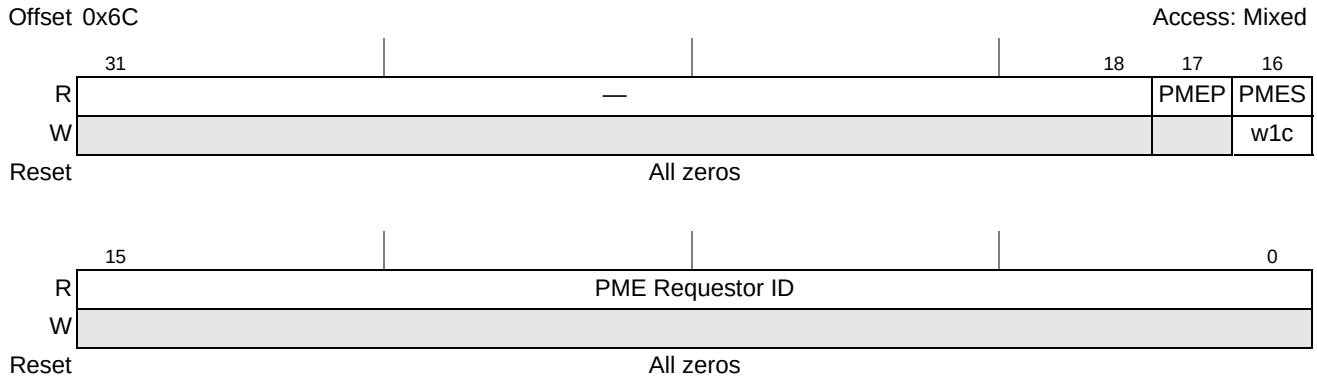


Figure 17-106. PCI Express Root Status Register

Table 17-108. 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.

17.4.1.8.18 PCI Express MSI Message Capability ID Register (EP Mode Only)—0x70

The PCI Express MSI message capability ID register is shown in **Figure 17-107**.

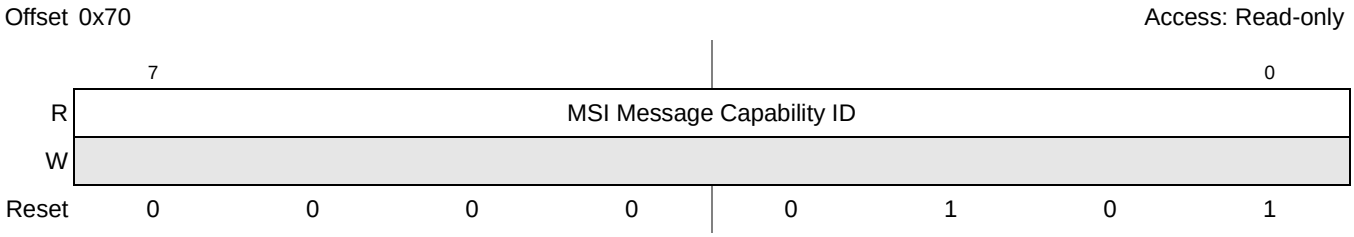


Figure 17-107. PCI Express Capability ID Register

Table 17-109. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	MSI Message Capability ID	MSI Message = 0x05

17.4.1.8.19 PCI Express MSI Message Control Register (EP Mode Only)—0x72

The PCI Express MSI message control register is shown in **Figure 17-108**.

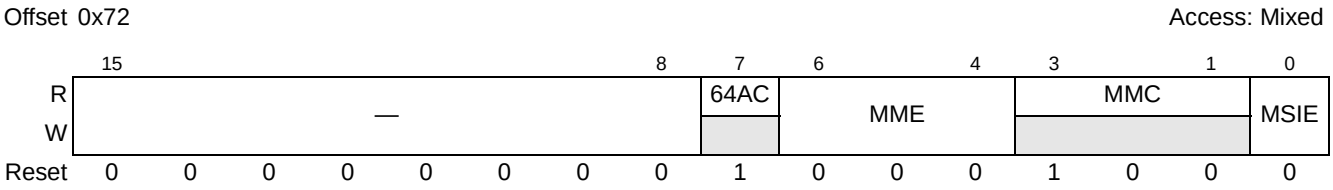


Figure 17-108. PCI Express MSI Message Control Register

Table 17-110. 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.

17.4.1.8.20 PCI Express MSI Message Address Register (EP Mode Only)—0x74

The PCI Express MSI message address register is shown in **Figure 17-109**.

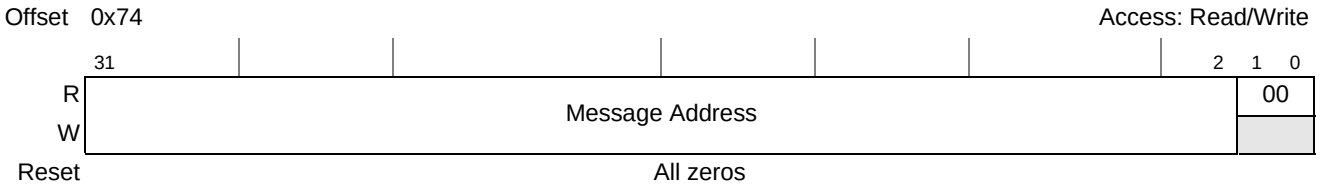


Figure 17-109. PCI Express MSI Message Address Register

Table 17-111. 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.

17.4.1.8.21 PCI Express MSI Message Upper Address Register (EP Mode Only)—0x78

The PCI Express MSI message upper address register is shown in **Figure 17-110**.

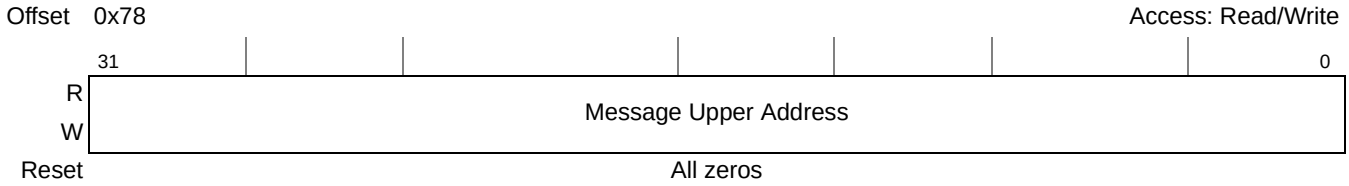


Figure 17-110. PCI Express MSI Message Upper Address Register

Table 17-112. PCI Express MSI Message Upper Address Register Field Description

Bits	Name	Description
31–0	Message Upper Address	System-specified message upper address

17.4.1.8.22 PCI Express MSI Message Data Register (EP Mode Only)—0x7C

The PCI Express MSI message data register is shown in **Figure 17-111**.

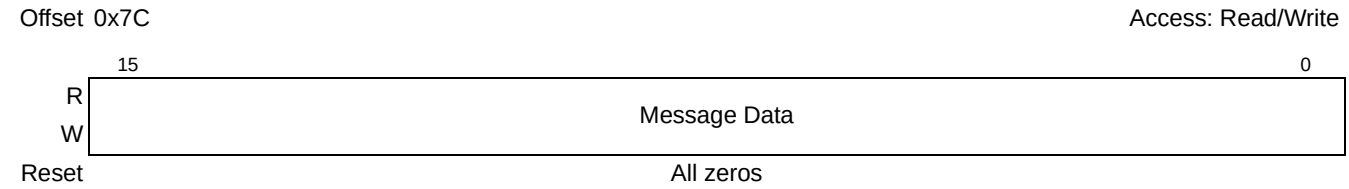


Figure 17-111. PCI Express MSI Message Data Register

Table 17-113. PCI Express MSI Message Data Register Field Description

Bits	Name	Description
15–0	Message Data	System-specified message.

17.4.1.9 PCI Express Extended Configuration Space

Reserved

		Address Offset (Hex)
PCI Compatible Configuration Header (See **Section 17.4.1.7**, *PCI Compatible Configuration Headers* for more information.)		000 03F
PCI-Compatible Device-Specific Configuration Space (See **Section 17.4.1.8**, *PCI Compatible Device-Specific Configuration Space* for more information.)		040 0FF
Next Capability Offset (NULL)/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 3FF
PCI Express Controller Internal CSRs		400 6FF
		700 FFF

Note: The PCI Express Controller Internal CSRs are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers returns all 0s.

Figure 17-112. PCI Express Extended Configuration Space

17.4.1.9.1 PCI Express Advanced Error Reporting Capability ID Register—0x100

The PCI Express advanced error reporting capability ID register is shown in **Figure 17-113**.

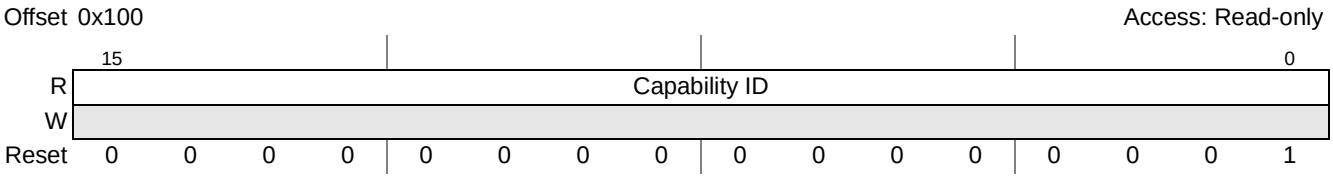


Figure 17-113. PCI Express Advanced Error Reporting Capability ID Register

Table 17-114. PCI Express Advanced Error Reporting Capability ID Register Field Description

Bits	Name	Description
15–0	Capability ID	Advanced error reporting capability = 0x0001

17.4.1.9.2 PCI Express Uncorrectable Error Status Register—0x104

The PCI Express uncorrectable error status register is shown in **Figure 17-114**.

Figure 17-114. PCI Express Uncorrectable Error Status Register

Table 17-115. 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.
13	FCPE	Flow control protocol error status.
12	PTLP	Poisoned TLP status.

Table 17-115. PCI Express Uncorrectable Error Status Register Field Description (Continued)

Bits	Name	Description
11–5	—	Reserved
4	DLPE	Data link protocol error status.
3–1	—	Reserved
0	TE	Training error status.

17.4.1.9.3 PCI Express Uncorrectable Error Mask Register—0x108

The PCI Express uncorrectable error mask register is shown in **Figure 17-115**.

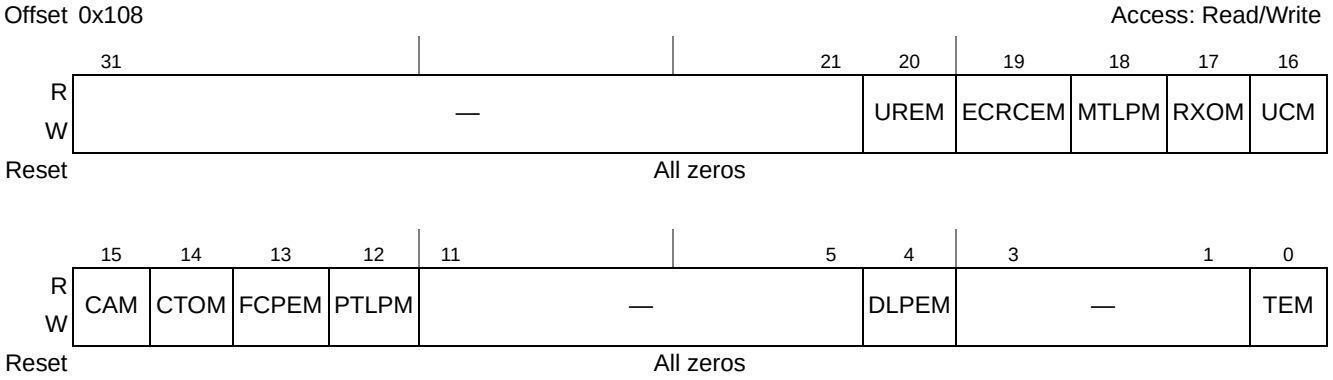


Figure 17-115. PCI Express Uncorrectable Error Mask Register

Table 17-116. 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.
3–1	—	Reserved
0	TEM	Training error mask.

17.4.1.9.4 PCI Express Uncorrectable Error Severity Register—0x10C

The PCI Express uncorrectable error severity register is shown in **Figure 17-116**.

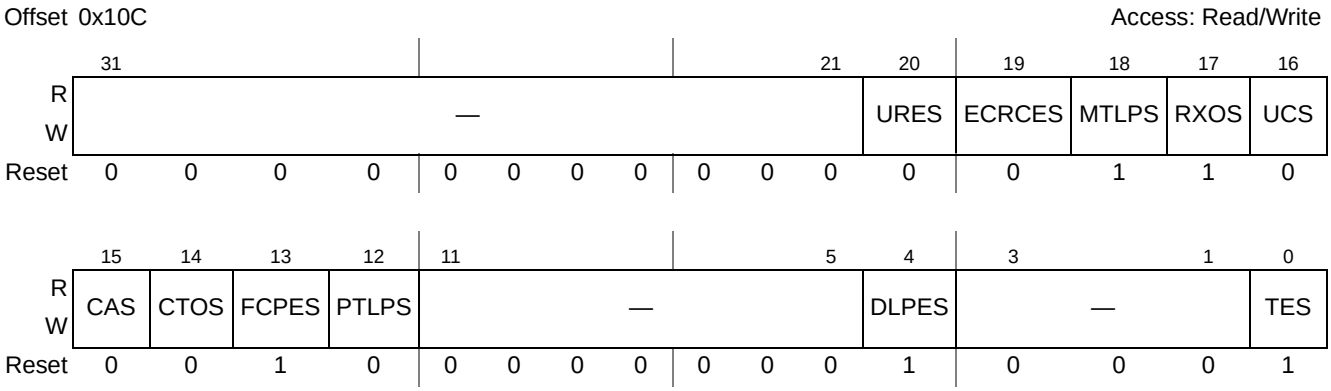


Figure 17-116. PCI Express Uncorrectable Error Severity Register

Table 17-117. 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.

17.4.1.9.5 PCI Express Correctable Error Status Register—0x110

The PCI Express correctable error status register is shown in **Figure 17-117**.

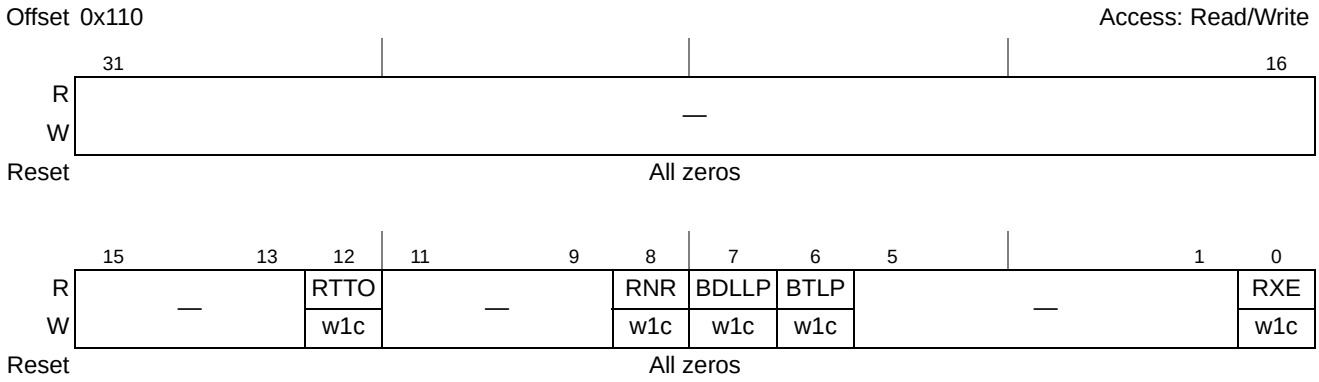


Figure 17-117. PCI Express Correctable Error Status Register

Table 17-118. PCI Express Correctable Error Status Register Field Description

Bits	Name	Description
31–13	—	Reserved
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

17.4.1.9.6 PCI Express Correctable Error Mask Register—0x114

The PCI Express correctable error mask register is shown in **Figure 17-118**.

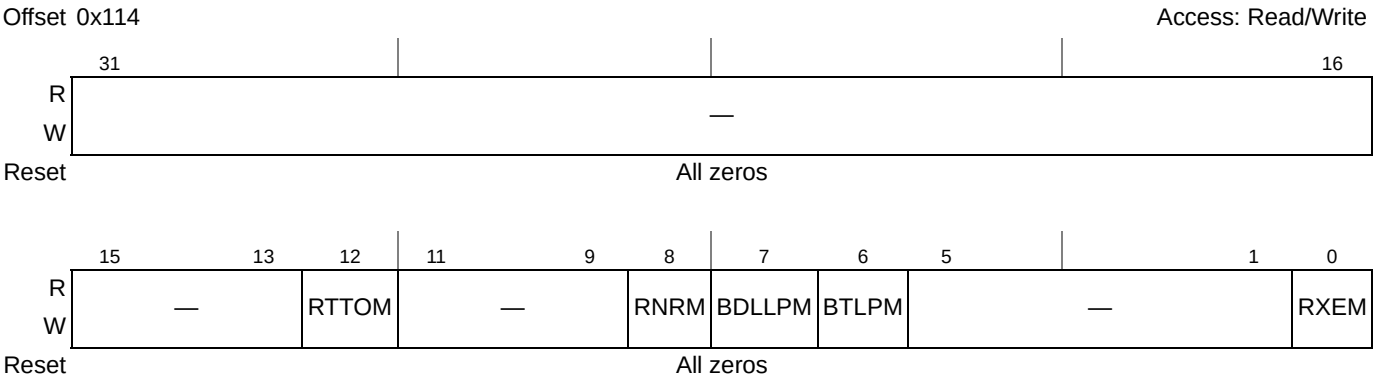


Figure 17-118. PCI Express Correctable Error Mask Register

Table 17-119. PCI Express Correctable Error Mask Register Field Description

Bits	Name	Description
31–13	—	Reserved
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

17.4.1.9.7 PCI Express Advanced Error Capabilities and Control Register—0x118

The PCI Express advanced error capabilities and control register is shown in **Figure 17-119**.

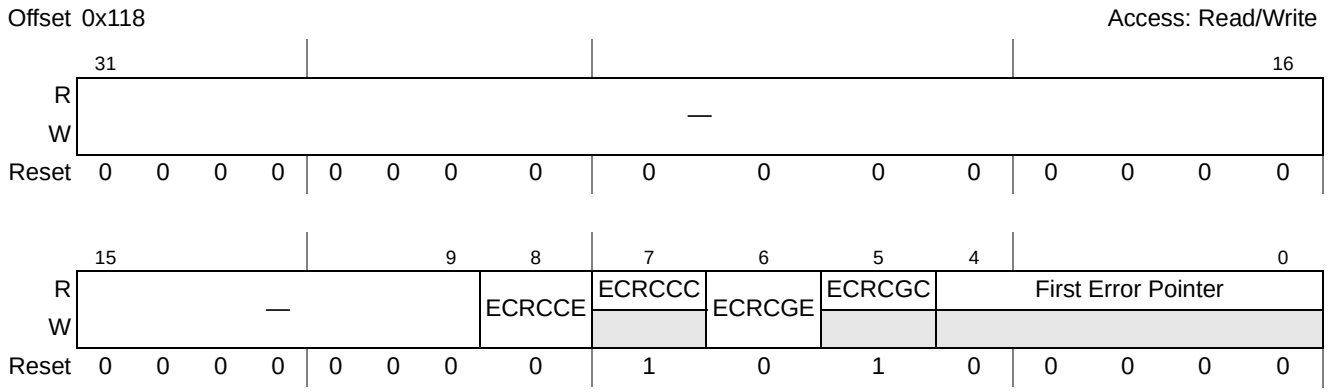


Figure 17-119. PCI Express Advanced Error Capabilities and Control Register

Table 17-120. 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.

17.4.1.9.8 PCI Express Header Log Register—0x11C–0x12B

The PCI Express header log register is shown in **Figure 17-120**.

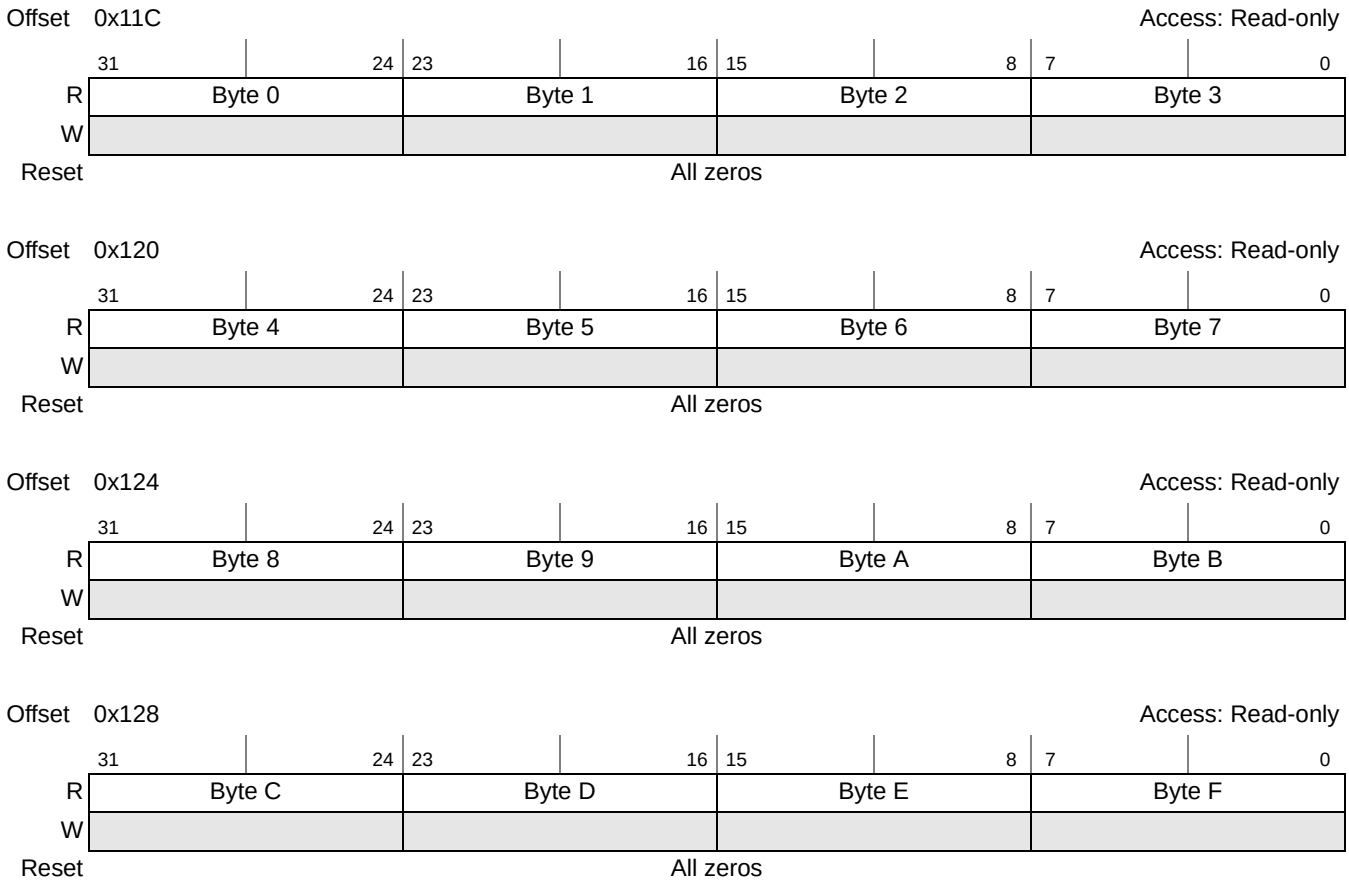


Figure 17-120. PCI Express Header Log Register

Table 17-121. PCI Express Header Log Register Field Description

Bits	Name	Description
127–0	Header Log	Header of TLP associated with error.

17.4.1.9.9 PCI Express Root Error Command Register—0x12C

The PCI Express root error command register is shown in **Figure 17-121**.

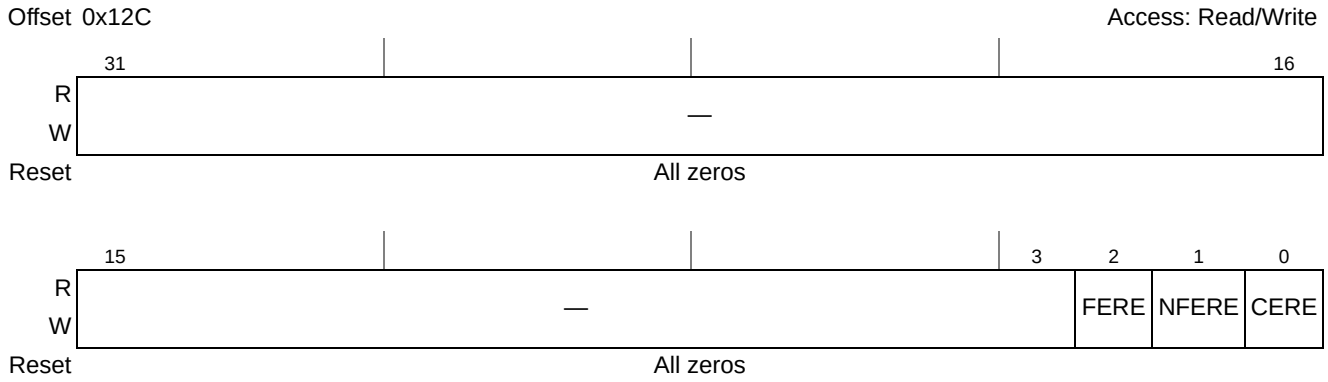


Figure 17-121. PCI Express Root Error Command Register

Table 17-122. 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

17.4.1.9.10 PCI Express Root Error Status Register—0x130

The PCI Express root error status register is shown in **Figure 17-122**.

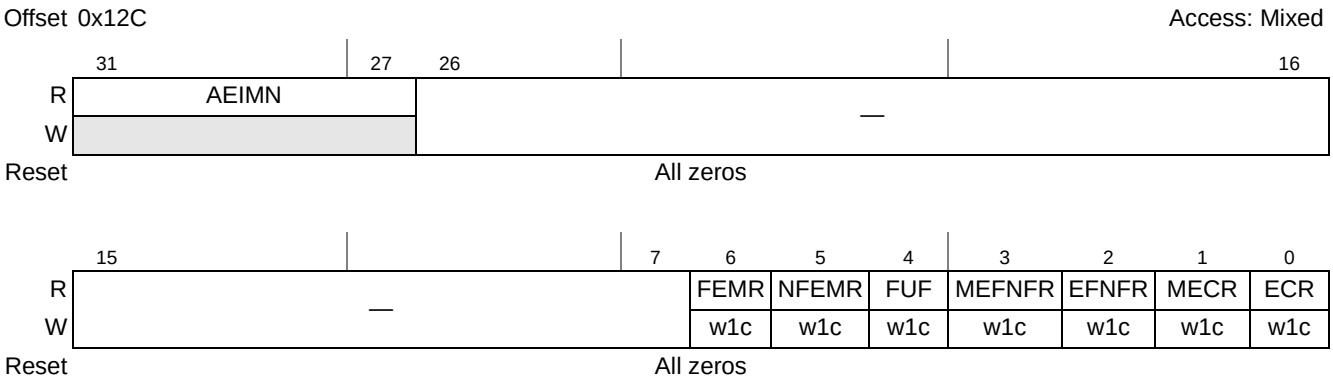


Figure 17-122. PCI Express Root Error Command Register

Table 17-123. PCI Express Root Error Command Register Field Description

Bits	Name	Description
31–27	AEIMN	Advanced error interrupt message number.
26–7	—	Reserved
6	FEMR	Fatal error messages received.
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.

17.4.1.9.11 PCI Express Correctable Error Source ID Register—0x134

The PCI Express correctable error source ID register is shown in **Figure 17-123**.

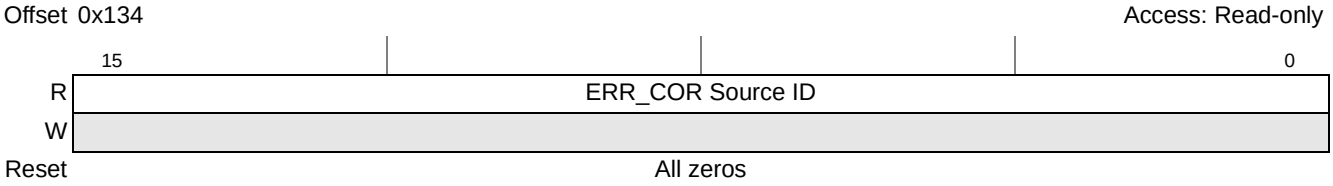


Figure 17-123. PCI Express Correctable Error Source ID Register

Table 17-124. 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.

17.4.1.9.12 PCI Express Error Source ID Register—0x136

The PCI Express error source ID register is shown in **Figure 17-124**.

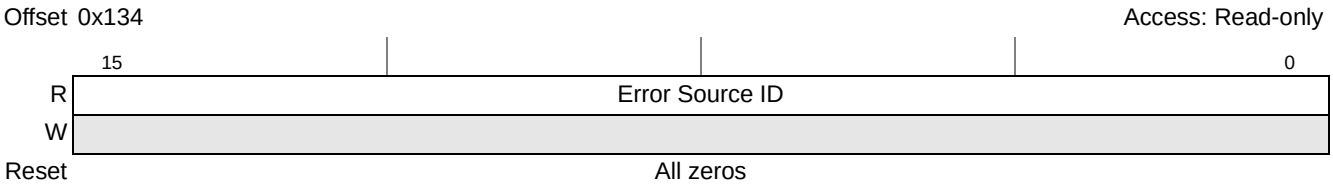


Figure 17-124. PCI Express Correctable Error Source ID Register

Table 17-125. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	Error Source ID	ERR_FATAL/NONFATAL source ID

17.4.1.9.13 LTSSM State Control Register—0x400

The PCI Express link training and status state machine (LTSSM) state control register, shown in **Figure 17-125**, is used to control the state transitions of the LTSSM in the MAC layer. This register is useful for debugging link training failures. During normal operation, all the bits of this register (except BYP_RXDET) must be cleared. The BYP_RXDET bit may be set as a temporary solution if there are issues with receiver detection in the PHY layer.

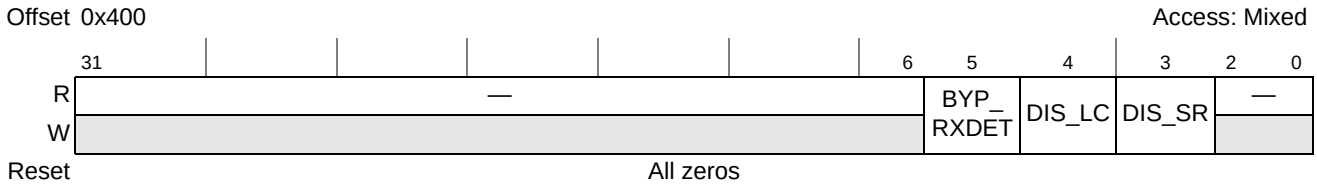


Figure 17-125. PCI Express LTSSM State Control Register (PEX_LTSSM_CONTROL)

The fields of the PCI Express LTSSM state control register are described in **Table 17-126**.

Table 17-126. PEX_LTSSM_CONTROL Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5	BYP_RXDET	Bypass receiver detect. 1 Receiver detect LTSSM state is bypassed. Useful if the receiver detection mechanism in the PHY layer has any issues during operation. 0 Normal operation. **Note:** This bit must be programmed before the PCI Express controller comes out of reset
4	DIS_LC	Disable LTSSM control. 1 Link training is skipped; the link comes up immediately. Useful when conducting loopback testing. This bit must be set to enabled bypass training, which allows the controller to perform lane reversal (transmitter lanes 0-1-2-3 connect to receiver lanes 3-2-1-0) if required; lane reversal is only valid for x4 mode. 0 Normal operation. **Note:** This bit must be programmed before the PCI Express controller comes out of reset.
3	DIS_SR	Disable scramble request. 1 Disable scramble request. 0 Normal operation. **Note:** This bit must be programmed before the PCI Express controller comes out of reset.
2–0	—	Reserved

17.4.1.9.14 LTSSM State Status Register—0x404

The PCI Express link training and status state machine (LTSSM) state status register, shown in **Figure 17-126**, provides detailed information about link training status. This register is useful for debugging link training failures.

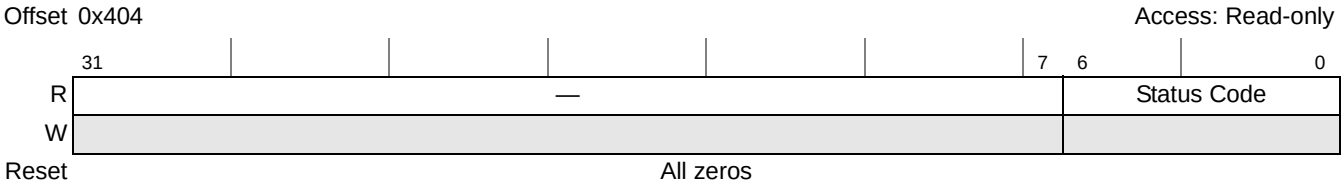


Figure 17-126. PCI Express LTSSM State Status Register (PEX_LTSSM_STAT)

The fields of the PCI Express LTSSM state status register are described in **Table 17-127**.

Table 17-127. PEX_LTSSM_STAT Field Descriptions

Bits	Name	Description
31–7	—	Reserved
6–0	Status code	Status code. See **Table 17-128** for encodings.

Table 17-128 provides the encodings for the status code field of the PEX_LTSSM_STAT register.

Table 17-128. PEX_LTSSM_STAT Status Codes

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
00	Detect quiet	27	TX L0s FTS; RX L0s FTS
01	Detect active (0)	28	L0 to L1 (0)
02	Detect active (1)	29	L0 to L1 (1)
03	Detect active (2)	2A	L1 entry
04	Polling active (0)	2B	L1 idle (0)
05	Polling active (1)	2C	L1 idle (1)
06	Polling config (0)	2D	L0 to L2 (0)
07	Polling config (1)	2E	L0 to L2 (1)
08	Polling compliance	2F	L2 entry
09	Configuration link width start (0)	30	L2 idle (0)
0A	Configuration link width start (1)	31	L2 idle (1)
0B	Configuration link width accept (0)	32	Recovery lock (0)
0C	Configuration link width accept (1)	33	Recovery lock (1)
0D	Configuration lane number wait (0)	34	Recovery lock (2)
0E	Configuration lane number wait (1)	35	Recovery cfg (0)
0F	Configuration lane number wait (2)	36	Recovery cfg (1)

Table 17-128. PEX_LTSSM_STAT Status Codes (Continued)

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
10	Configuration lane number wait (3)	37	Recovery idle (0)
11	Configuration lane number accept	38	Recovery idle (1)
12	Configuration complete (0)	39	Recovery to configuration
13	Configuration complete (1)	3A	Recovery cfg to configuration
14	Configuration idle (0)	3F	L0 no training
15	Configuration idle (1)	7F	Detect quiet EI
16	L0	49	Configuration link width start—RC
17	TX L0; RX L0s entry	4A	Configuration link width accept—RC
18	TX L0; RX L0s idle	4B	Configuration lane number wait—RC
19	TX L0; RX L0s fast training sequence (FTS)	4C	Configuration lane number accept—RC
1A	TX L0s entry (0); RX L0	60	Loopback slave active (0)
1B	TX L0s entry (0); RX L0s idle	61	Loopback slave active (1)
1C	TX L0s entry (0); RX L0s FTS	62	Loopback slave exit
1D	TX L0s entry (1); RX L0	68	Hot reset (0)
1E	TX L0s entry (1); RX L0s idle	69	Hot reset (1)
1F	TX L0s entry (1); RX L0s FTS	6A	Hot reset (0)—RC
20	TX L0s idle; RX L0	6B	Hot reset (1)—RC
21	TX L0s idle; RX L0s entry	75	Disabled (0)
22	TX L0s idle; RX L0s idle	71	Disabled (1)
23	TX L0s idle; RX L0s FTS	72	Disabled (2)
24	TX L0s FTS; RX L0	73	Disabled (3)
25	TX L0s FTS; RX L0s entry	74	Disabled (4)
26	TX L0s FTS; RX L0s idle	78	L0 to L1/L2—RC

17.4.1.9.15 PCI Express ACK Replay Timeout Register (PEX_ACK_REPLAY_TIMEOUT)—0x434

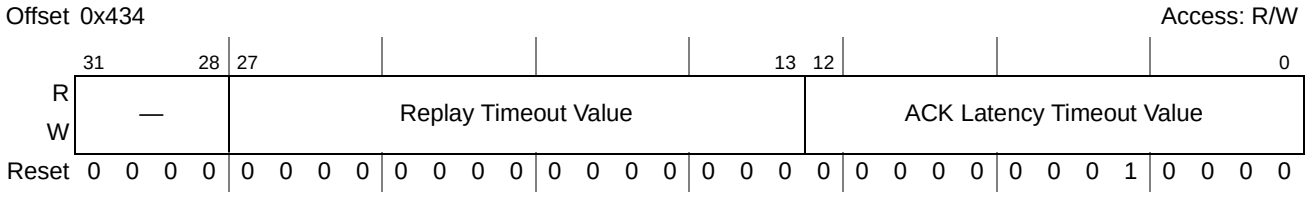


Figure 17-127. PCI Express ACK Replay Timeout Register (PEX_ACK_REPLAY_TIMEOUT)

The fields of the PCI Express ACK replay timeout register are described in **Table 17-130**.

Table 17-129. PEX_ACK_REPLAY_TIMEOUT Field Descriptions

Bits	Name	Description
31–28	—	Reserved
27–13	Replay_ Timout_Value	Timeout value to wait for reception of ACK DLLP from link side by DLL before re-transmitting TLPs. Protocol specifies this value in symbol times for various combinations of max-payload size and negotiated linkwidth . Appropriate value should be chosen and programmed into this register. This value can be calculated as (Symbol_time * CORE_CLK in MHz /250+ Rx L0s adjustment offset in terms of core clk cycles).
12–0	ACK_ Latency_ Timout_Value	Timeout value to force transmission of ACK DLLP by DLL after a TLP is received. Protocol specifies this value in symbol times for various combinations of max-payload size and negotiated linkwidth. Appropriate value should be chosen and programmed into this register. This value can be calculated as (Symbol_time * CORE_CLK in MHz /250 + Tx L0s adjustment offset in terms of core clk cycles).

This register is used to program timeout values for ACK DLLP transmission and reception in DLL. ACK receive timeout is termed Replay timeout because TLPs are re-transmitted after this timeout. Both values should be in terms of CORE clk cycles and must be set based on Max-Payload-size and operating link width as specified in the protocol (P-140 and P149). ACK time-out also depends upon ASPM L0s enabling for the Tx link of the device.

The CSR implements a look-up table for automatic updating of ACK and replay time-out values, based on max-payload size, link_width, and ASPM L0s enabled information. So, the reset value of this register is difficult to check. The automatic update of these values is disabled upon the first write to this register by UL because it is assumed that UL takes care of the update based on the above factors. The macros related to the look-up table are available in the include file gpex_csr.vh in the include directory. This file is generated by VPP parsing gpex_csr.vhpp file.

17.4.1.9.16 PCI Express Controller Core Clock Ratio Register—0x440

The PCI Express controller core clock ratio register, shown in **Figure 17-128**, is used to program the ratio of the actual PCI Express controller clock frequency to the default controller core frequency (333 MHz). This is required only when a PCI Express controller clock frequency other than the default 333 MHz has to be used.

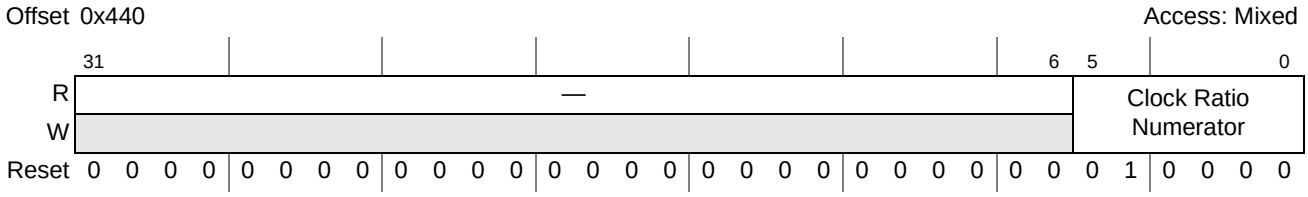


Figure 17-128. PCI Express IP Block Core Clock Ratio Register (PEX_GCLK_RATIO)

The fields of the PCI Express IP block core clock ratio register are described in **Table 17-130**.

Table 17-130. PEX_GCLK_RATIO Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5–0	Clock Ratio Numerator	The numerator of the ratio of the actual PCI Express controller clock frequency used to the default core clock frequency of 333 MHz. The denominator of the ratio is fixed at 16. The default value of this register is 0x10 (16 decimal), which corresponds to a ratio of 1:1 (or 16/16)

As an example of programming PEX_GCLK_RATIO, consider the case where the actual PCI Express controller clock is 250 MHz, the ratio of the actual clock to the default clock (333 MHz) is 3:4. that is, the default core clock has to be multiplied by the ratio (3/4, which is equivalent to 12/16). So the register has to be programmed with the decimal numerator value 12 or 0x0000_000C.

17.4.1.9.17 PCI Express Power Management Timer Register—0x450

The PCI Express power management timer register, shown in **Figure 17-129**, is used to program the time-in values for entering L0s and L1 power management states.

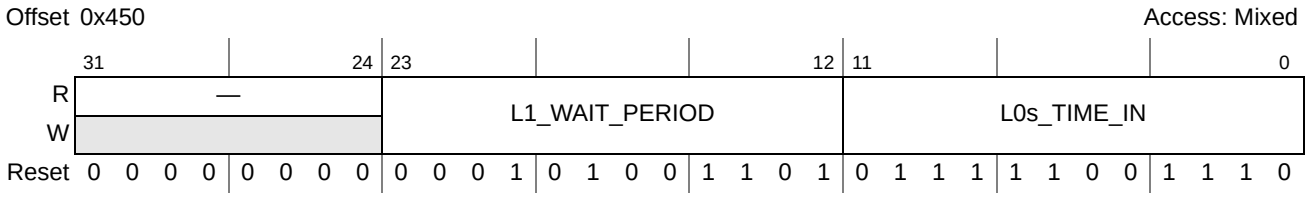


Figure 17-129. PCI Express Power Management Timer Register (PEX_PM_TIMER)

The fields of the PCI Express power management timer register are described in **Table 17-131**.

Table 17-131. PEX_PM_TIMER Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–12	L1_WAIT_PERIOD	Wait period (in PCI Express controller core clock cycles) before entering L1 power state after all functions are in a non-D0 power state. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The time value must be less than 2 µs; the default value (0x14D) is 1 µs for the default clock frequency of 333 MHz.
11–0	L0s_TIME_IN	Time in value (in PCI Express controller core clock cycles) for entering L0s power state. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The maximum time value is 7 µs; the default value (0x7CE) is 6 µs for the default clock frequency of 333 MHz.

17.4.1.9.18 PCI Express PME Time-Out Register (EP-Mode Only)—0x454

The PCI Express PME time-out register, shown in **Figure 17-130**, is used to program the time-out value that the controller uses before re-sending a PME message to the host. If PME is requested by a function and the host does not clear the associated PME_STAT bit even after this time-out has expired, the PME message is sent again to the host by the PCI Express controller. This register is supported only for EP mode.

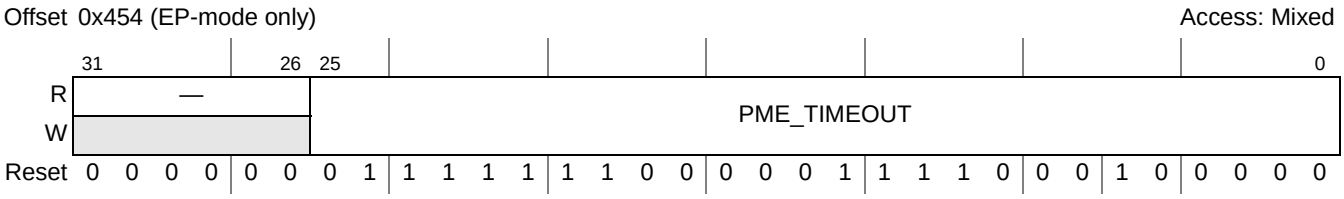


Figure 17-130. PCI Express PME Time-Out Register (PEX_PME_TIMEOUT)

The fields of the PCI Express PME time-out register are described in **Table 17-131**.

Table 17-132. PEX_PME_TIMEOUT Field Descriptions

Bits	Name	Description
31–26	—	Reserved
25–0	PME_TIMEOUT	The PME time-out value specifies the interval before PME messages are resent by the controller, provided the PME_STAT bit in the PCI Express power management status and control register (offset 0x48) is not cleared by the host. The value for PME_TIMEOUT is specified in terms of PCI Express controller core clock cycles. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The minimum time value is 100 ms; the default value (0x1FC1E20) is 100 ms for the default clock frequency of 333 MHz.

17.4.1.9.19 PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478

The PCI Express subsystem vendor ID update register, shown in **Figure 17-131**, is used to set the values for the Subsystem ID and Subsystem Vendor ID registers in the Type 0 configuration header.

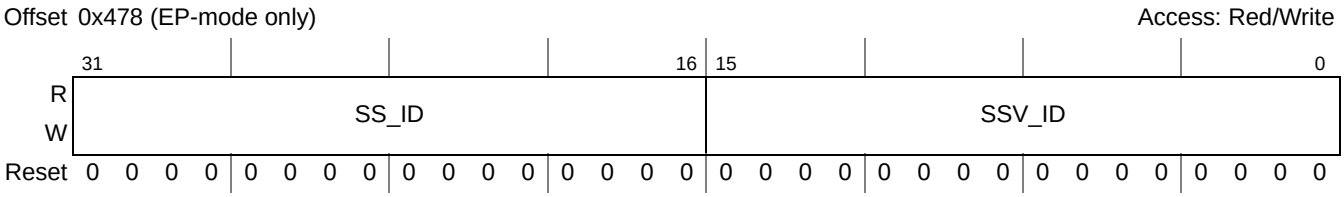


Figure 17-131. PCI Express Subsystem Vendor ID Update Register (PEX_SSVID_UPDATE)

The fields of the PCI Express subsystem vendor ID update register are described in **Table 17-133**.

Table 17-133. PEX_SSVID_UPDATE Field Descriptions

Bits	Name	Description
31–16	SS_ID	Subsystem ID [15–0] value
15–0	SSV_ID	Subsystem vendor ID [15–0] value

When used as an endpoint, the controller's initialization software programs the desired subsystem ID and subsystem vendor ID values in PEX_SSVID_UPDATE before setting the CFG_READY bit in the PEX_CFG_READY register (see **Section 17.4.1.9.20**, *Configuration Ready Register—0x4B0*, on page 17-119). That way, when the host begins system enumeration, the correct values will be present in the Type 0 configuration header.

17.4.1.9.20 Configuration Ready Register—0x4B0

The PCI Express configuration ready register, shown in **Figure 17-132**, is used to indicate configuration complete status to the transaction layer. The transaction layer handles configuration requests from external hosts only after the CFG_READY bit is set. All the configuration requests received from external hosts before the CFG_READY bit is set are completed with configuration request retry status (CRS). The CFG_READY bit in this register should be set after all relevant configuration registers have been programmed. This makes sure the external host reads the correct capability advertisements during enumeration.

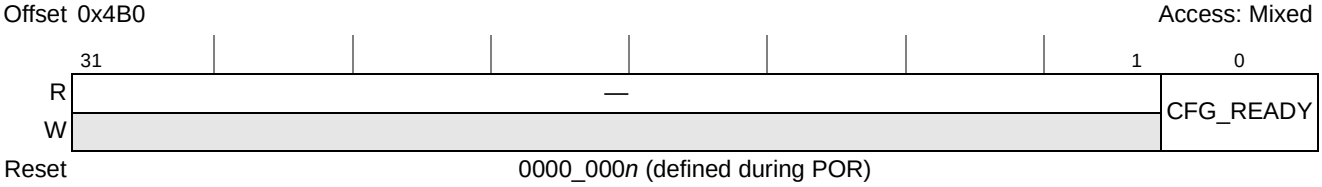


Figure 17-132. PCI Express Configuration Ready Register (PEX_CFG_READY)

The fields of the PCI Express configuration ready register are described in **Table 17-134**.

Table 17-134. PEX_CFG_READY Field Descriptions

Bits	Name	Description
31–1	—	Reserved
0	CFG_READY	Configuration ready 1 The transaction layer will accept inbound configuration requests. 0 The transaction layer will respond to all inbound configuration requests with retry (CRS) Note that the reset state of this bit is determined during POR.

17.4.1.9.21 PME_To_Ack Timeout Register (RC-Mode Only)—0x590

The PCI Express PME_To_Ack timeout register, shown in **Figure 17-133**, is used to program the timeout value for a PME_To_Ack message response in terms of PCI Express controller core clock cycles.

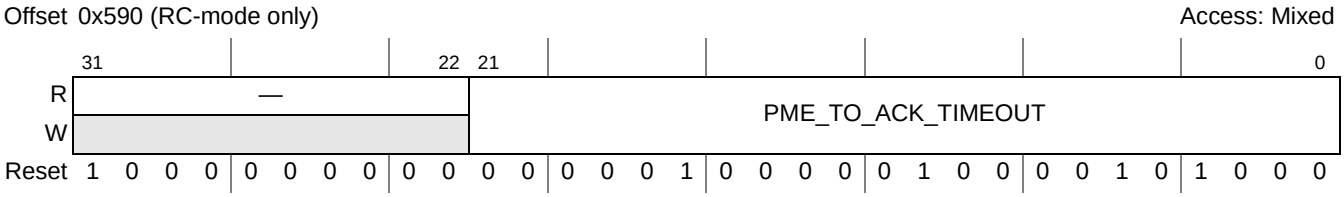


Figure 17-133. PCI Express PME_To_Ack Timeout Register (PEX_PME_TO_ACK_TOR)

The fields of the PCI Express PME_To_Ack timeout register are described in **Table 17-135**.

Table 17-135. PEX_PME_TO_ACK_TOR Field Descriptions

Bits	Name	Description
31–22	—	Reserved
21–0	PME_TO_ACK_TIMEOUT	After a PME_Turn_Off message is broadcast by the RC, the power management module waits for the duration of the PME_To_Ack timeout interval to receive a PME_To_Ack message from the downstream device. If the Ack message is not received within this interval, the power manager indicates that it is safe to switch off power, since timeout has occurred. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The recommended timeout duration is 1 ms to 10 ms to make sure that the downstream devices get enough time to prepare for power-off condition.

17.4.1.9.22 Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0

The PCI Express secondary status interrupt mask register, shown in **Figure 17-134**, can be used to disable sideband interrupt generation when error bits in the PCI Express secondary status register are set. See **Section 17.4.1.7.29**, *PCI Express Secondary Status Register—0x1E*, on page 17-83 for more information. By default, interrupt generation due to secondary status errors is disabled.

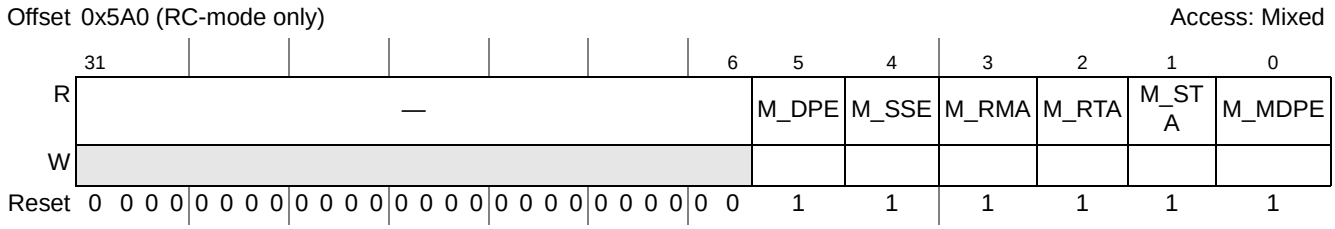


Figure 17-134. PCI Express PCI Interrupt Mask Register (PEX_SS_INTR_MASK)

The fields of the PCI Express secondary status interrupt mask register are described in **Table 17-136**.

Table 17-136. PEX_SS_INTR_MASK Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5	M_DPE	Mask detected parity error
4	M_SSE	Mask signaled system error
3	M_RMA	Mask received master abort
2	M_RTA	Mask received target abort
1	M_STA	Mask signaled target abort
0	M_MDPE	Mask master data parity error

17.4.1.9.23 Gen2 Control Register

The PEX Gen2 control register, shown in **Figure 17-135**, can be used to control the MAC in Gen2 device.

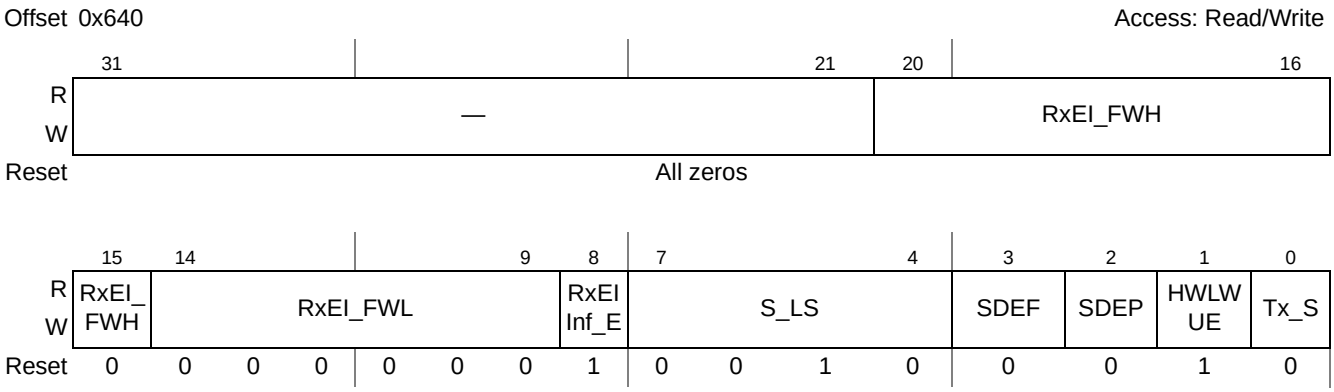


Figure 17-135. PEX Gen2 Control Register (PEX_GEN2_CONTROL)

The fields of the PEX Gen2 control register are described in **Table 17-137**.

Table 17-137. PEX_GEN2_CONTROL Field Descriptions

Bits	Name	Description
31–21	—	Reserved
20–15	RxEI_FWH	RxEI filter width high
14–9	RxEI_FWL	RxEI filter width low
8	RxEI_Inf_E	RxEI inferring enable
7–4	S_LS	Supported link speeds
3	SDEF	Select de-emphasis forced
2	SDEP	Select de-emphasis preferred
1	HWLWUE	Hardware link width upconfiguration enable
0	Tx_S	Transmit swing

QUICC Engine Subsystem 18

The QUICC Engine subsystem is a versatile RISC-based communication processor that supports multiple external interfaces and protocols independently from the core processor(s) in an integrated processing device. This allows the cores to execute the data processing code and be relieved from the data transfer and handling overhead. This chapter provides an overview of the QUICC Engine subsystem components used in the MSC8156E, which include the following:

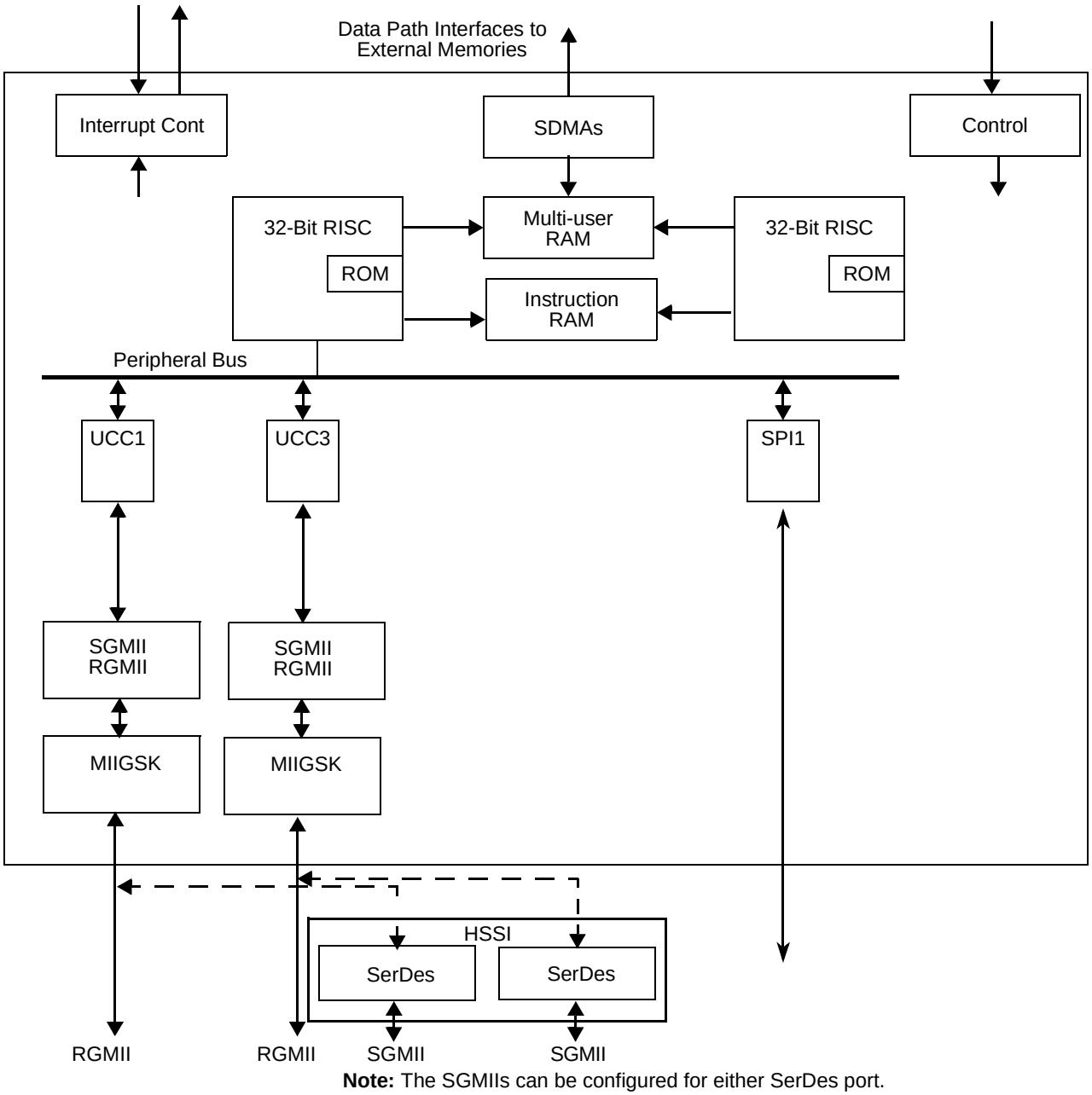
- Dual RISC engines with
  - Internal interfaces to the core and peripherals
  - Parameter RAM
  - Buffer descriptors
  - Multithreading operation
  - Serial numbers (SNUMs)
  - Instruction RAM (IRAM)
- Serial DMA controller
- Clocking
  - Signal multiplexing
  - Baud-rate generation
- Dedicated interrupt controller
- Two programmable Unified Communication Controllers (UCCs), each of which provides dedicated support for an Ethernet controller for RGMII/SGMII interfaces
- Serial peripheral interface (SPI)

Note: Detailed information about QUICC Engine functionality and programming is provided in the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM). Refer to that manual for all the functional and programming details

Note: If the QUICC Engine subsystem is not used for booting, the user must clear the QUICC Engine DRAM before using it to ensure correct operation.

18.1 Overview

Figure 18-1 shows an architectural diagram of the QUICC Engine subsystem. The UCCs group is controlled by a RISC engine. A common multi-user RAM is used to store parameters for RISC engines. Each RISC has a ROM associated with it that contains the code image. The instruction RAM is used to run RISC code from the RAM. The internal clocks (RCLK/TCLK) for each UCC can be programmed to use either an external or internal source. The rate of these clocks can be up to one-half of the QUICC Engine subsystem clock frequency. However, the ability of an interface to support a sustained bit stream depends on the protocol settings and other factors.



Note: The SGMIIs can be configured for either SerDes port.

Figure 18-1. QUICC Engine Subsystem Architectural Block Diagram

18.2 RISC Processors

The two 32-bit RISC processors reside on a bus separate from the SC3850 cores, and can, therefore, perform tasks independently from the DSP cores. The RISC processors handle lower-layer communications tasks and DMA control, freeing the DSP cores to handle higher-layer activities. The RISC processors work with the UCCs to implement user-programmable protocols and manage the serial DMA (SDMA) channels that transfer data between the I/O channels and memory. The RISC processor architecture and instruction set are optimized for data communications and data processing required by many wire-line and wireless communications standards. The SC3850 DSP cores issues commands to the RISC processors by writing to the QUICC Engine Command Register (CECR), which is described in **Section 18.9**. The RISC processors execute the commands to ensure synchronization with other tasks running on the QUICC Engine subsystem. The DSP core sets the CECR[FLG] bit when it issues a command, and the QUICC Engine subsystem clears the FLG after execution, indicating to the DSP core that it is ready for the next command. Subsequent commands to the CECR can be given only after FLG is clear. The software reset command may be issued by setting RST even if the FLG bit is set. The CECR rarely needs to be accessed. For example, to terminate the transmission of a frame without waiting until the end, a STOP TX command is issued through the CECR. The worst-case command execution latency is 200 clocks and the typical command execution latency is about 40 clocks. The RISC processors give SDMA commands to the SDMA. RISC processor features include:

- One QUICC Engine clock cycle per instruction
- 32-bit instruction object code
- Code execution from internal ROM or RAM
- 32-bit ALU data path
- 64-bit multi-port RAM access
- Optimized for communications processing
- DMA bursting of serial data to/from external memory

18.2.1 SC3850 Core Interface

The RISC processors communicate with the SC3850 cores in several ways:

- Many parameters are exchanged through the multi-port RAM.
- The RISC processors can execute special commands issued by the SC3850 cores. These commands should be issued only in special situations such as exceptions or error recovery.
- The RISC processor generates interrupts through the interrupt controller.
- The SC3850 cores can read the QUICC Engine subsystem status/event registers at any time.

18.2.2 Peripheral Interface

The RISC processors use the peripheral bus to communicate with all of its UCCs. Each controller has separate receive and transmit 192-byte FIFOs.

18.2.3 Parameter RAM

The QUICC Engine subsystem maintains a section in the multi-user RAM that contains the associated parameter values for the operation of the UCCs and SPI. Each peripheral has an associated page in the parameter RAM. The exact definition of the parameter RAM for each peripheral is in the protocol descriptions in the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM). The size of the parameter RAM page differs from protocol to protocol. The minimum size of the parameter RAM page assigned to any protocol is 64 bytes. The base address of the parameter RAM page must be aligned to 64 bytes.

Some parameter RAM values must be initialized before the UCC is enabled. The QUICC Engine subsystem initializes or writes other values. Once initialized, most parameter RAM values need not be accessed by user software, because most activity centers around the TxBDs and RxBDs rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- You can read the parameter RAM at any time.
- You can write to the Tx parameter RAM only when the transmitter is disabled (that is, after a STOP TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command).
- You can write to the Rx parameter RAM only when the receiver is disabled. The CLOSE RX BD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.

After reset, the QUICC Engine subsystem initializes the base addresses of the parameter RAM pages for all the UCCs to default values. You can override the default values by issuing the ASSIGN PAGE command to the QUICC Engine subsystem. The advantage of using the ASSIGN PAGE command is that you can arrange the parameter RAM area efficiently (with no unused areas) according to the specific peripherals/protocols used in your system.

Table 18-1 lists the default values of the parameter RAM pages base addresses assigned by the QUICC Engine subsystem to the different peripherals.

Table 18-1. Default Parameter RAM Base Addresses

Address Offset	Peripheral	Size (Bytes)
0x8400	UCC 1 (Rx and Tx)	256
0x8600	UCC 3 (Rx and Tx)	256
0x8900	SPI (Rx and Tx)	128

18.2.4 Buffer Descriptors (BDs)

If you are programming the UCCs, you need to know how the controllers use buffer descriptors to define buffer allocation. A buffer descriptor (BD) contains the essential information about each buffer in memory. Each buffer is referenced by a BD that can reside anywhere in system memory. These BDs are split between all UCCs.

Each 64-bit BD has the structure shown in **Figure 18-2**. This structure is common to all UCCs. A receive buffer descriptor (RxBD) table and a transmit buffer descriptor (TxBD) table are associated with each controller. Each table can have multiple BDs.

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	Status and Control															
0x2	Data Length															
0x4	High-Order Buffer Pointer															
0x6	Low-Order Buffer Pointer															

Figure 18-2. Buffer Descriptor Structure

In this discussion, the BD and field values use the following convention:

```
BD.field
```

Table 18-2 shows the possible BD and field naming conventions. Bit names in `RxBD.bd_cstat` and `TxBD.bd_cstat` use the following convention:

```
BD.bd_cstat.bit
```

Table 18-2. Buffer Descriptor Name Convention

BD	Field	Example
RxBD/TxBD	bd_cstat	`TxBD.bd_cstat`.R refers to the ready bit in the TxBD's status and control field.
	bd_length	`RxBD.bd_length` refers to RxBD data length field.
	bd_addr	`RxBD.bd_addr` refers to RxBD buffer pointer field.

The structural elements of a buffer descriptor are defined as follows:

- *Status and control*. The 16-bit value at `offset+0x0`, which contains status and control bits that control and report status information on the data transfer. The RISC processor updates the status bits after the buffer is sent or received. Only this field differs for each protocol. Refer to the relevant chapter in this manual for each protocol `RxBD.bd_cstat` and `TxBD.bd_cstat` bit description.
- *Data length*. The 16-bit value at `offset+0x2`, which contains the number of bytes sent or received.

- *RxBD data length*. The number of bytes the RISC processor writes into the RxBD buffer once the BD closes. The RISC processor updates this field after the received data is placed into the buffer and the buffer is closed. You do not need to initialize this field. In frame-based protocols, `RxBD.bd_length` contains the total frame length including CRC bytes. If a received frame length, including CRC, is an exact multiple of the parameter RAM maximum receive buffer length MRBLR, the last buffer holds no actual data but the associated BD contains the total frame length.
- *TxBD data length*. The number of data bytes the controller needs to transmit from its buffer. The RISC processor never modifies this field. This field needs to be initialized by the user.
- *Buffer pointer*. The 32-bit data at `offset+0x4`, which points to the beginning of the buffer in internal or external memory.
- *RxBD buffer pointer*. The buffer pointer value must be a multiple of four to be word-aligned.
- *TxBD buffer pointer*. The buffer pointer value can be even or odd.

18.2.5 Multithreading

The Ethernet Controllers are able to processes frames or cells at high bit rates (gigabit Ethernet and above OC-3 nominal rates). In order to achieve these bit rates the UCC receiver and the UCC transmitter are able to process multiple frames/cells simultaneously at any given time. This is implemented with the multithreading mechanism. Each thread processes a different frame/cell. The multithreading processing mechanism comprises three components: Distributor, Threads and in some cases Terminator. **Figure 18-3** shows a high-level diagram of the multithreading architecture.

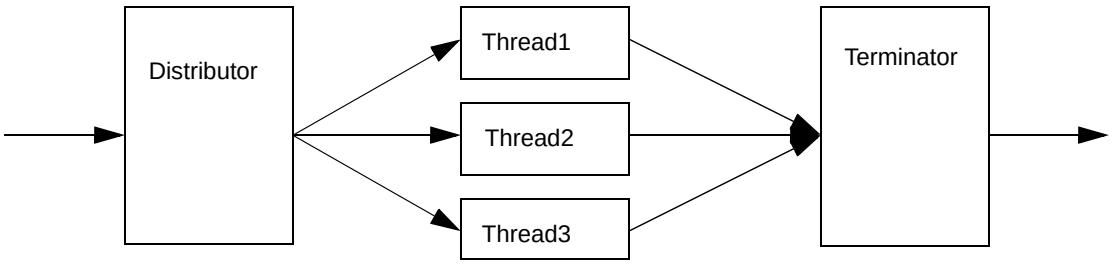


Figure 18-3. Multi-Threading Processing Mechanism

Each one of the components (distributor, threads and terminator) has an ID number associated with it, referred to as its Serial Number (SNUM). The distributor SNUM is always the SNUM of the UCC receiving or transmitting the data. Each one of the transmitter and receiver threads has its own parameter RAM located in the multi-port RAM. The user software initializes the values for the SNUM and the pointer of the parameter RAM base address at initialization time.

Note: See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM).for specific interface configuration details.

18.2.6 Serial Numbers (SNUMs)

Each peripheral has a unique serial number (SNUM) associated with it. Threads, which are used by the multithreading mechanisms described in **Section 18.2.5**, *Multithreading* also have a unique serial number. There are two cases for which you should specify the SNUM:

1. When issuing the ASSIGN PAGE command.
2. When initializing the multithreading mechanism.

Table 18-3 lists the Serial Numbers (SNUMs) used to program the multithreading options in the UCCs for the Ethernet controllers. The component column indicates the peripheral or thread name for each SNUM.

Table 18-3. SNUM Table

SNUM	Component	SNUM	Component	SNUM	Component	SNUM	Component
0x00	UCC1 TX	0x40	—	0x80		0xC0	Reserved
0x01	UCC1 RX	0x41	—	0x81		0xC1	Reserved
0x04	Thread16	0x44	—	0x84	—	0xC4	—
0x05	Thread17	0x45	—	0x85	—	0xC5	—
0x08	—	0x48		0x88	Thread0	0xC8	Thread8
0x09	—	0x49		0x89	Thread1	0xC9	Thread9
0x0C	Thread18	0x4C	—	0x8C	—	0xCC	—
0x0D	Thread19	0x4D	—	0x8D	—	0xCD	—
0x10	Reserved	0x50	Reserved	0x90	Reserved	0xD0	—
0x11	Reserved	0x51	Reserved	0x91	Reserved	0xD1	—
0x14	Thread20	0x54	—	0x94	—	0xD4	—
0x15	Thread21	0x55	—	0x95	—	0xD5	—
0x18	—	0x58	—	0x98	Thread2	0xD8	Thread10
0x19	—	0x59	—	0x99	Thread3	0xD9	Thread11
0x1C	Thread22	0x5C	—	0x9C	—	0xDC	—
0x1D	Thread23	0x5D	—	0x9D	—	0xDD	—
0x20	UCC3 TX	0x60	Reserved	0xA0	Reserved	0xE0	
0x21	UCC3 RX	0x61	Reserved	0xA1	Reserved	0xE1	
0x24	Thread24	0x64	—	0xA4	—	0xE4	—
0x25	Thread25	0x65	—	0xA5	—	0xE5	—
0x28	—	0x68	—	0xA8	Thread4	0xE8	Thread12
0x29	—	0x69	—	0xA9	Thread5	0xE9	Thread13
0x2C	Thread26	0x6C	—	0xAC	—	0xEC	—
0x2D	Thread27	0x6d	—	0xAD	—	0xED	—
0x30	Reserved	0x70	Reserved	0xB0	Reserved	0xF0	TIMER
0x31	Reserved	0x71	Reserved	0xB1	Reserved	0xF1	Lowest
0x34	Thread28	0x74	—	0xB4	—	0xF4	—
0x35	Thread29	0x75	—	0xB5	—	0xF5	—
0x38	—	0x78	—	0xB8	Thread6	0xF8	Reserved
0x39	—	0x79	—	0xB9	Thread7	0xF9	Reserved
0x3C	Reserved	0x7C	—	0xBC	—	0xFC	—
0x3D	Reserved	0x7D	—	0xBD	—	0xFD	—

Note: See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details on how to use the SNUM with the ASSIGN PAGE command.

18.2.7 IRAM

The QUICC Engine subsystem includes 48 KB of IRAM that can be used to store processing code for the RISC processors. The IRAM can be split into two 24-KB areas assigned individually to each of the two RISC processors, or it can be shared by both processors as one contiguous 48-KB memory space. Access is through the IRAM address register (IADDR) using the IRAM data register (IDR). See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

18.3 Serial DMA Controller

The QUICC Engine subsystem has one physical serial DMA (SDMA) channel that interfaces with the internal MBus. The QUICC Engine subsystem implements two dedicated virtual SDMA channels for each peripheral, one for the receiver and one for the transmitter. Additional virtual channels are available for general purpose accesses.

18.3.1 Data Paths

Figure 18-4 is a simplified block diagram that shows the data paths in the MSC8156E. The MSC8156E may be implemented with one DDR bus (32 or 64 bit data).

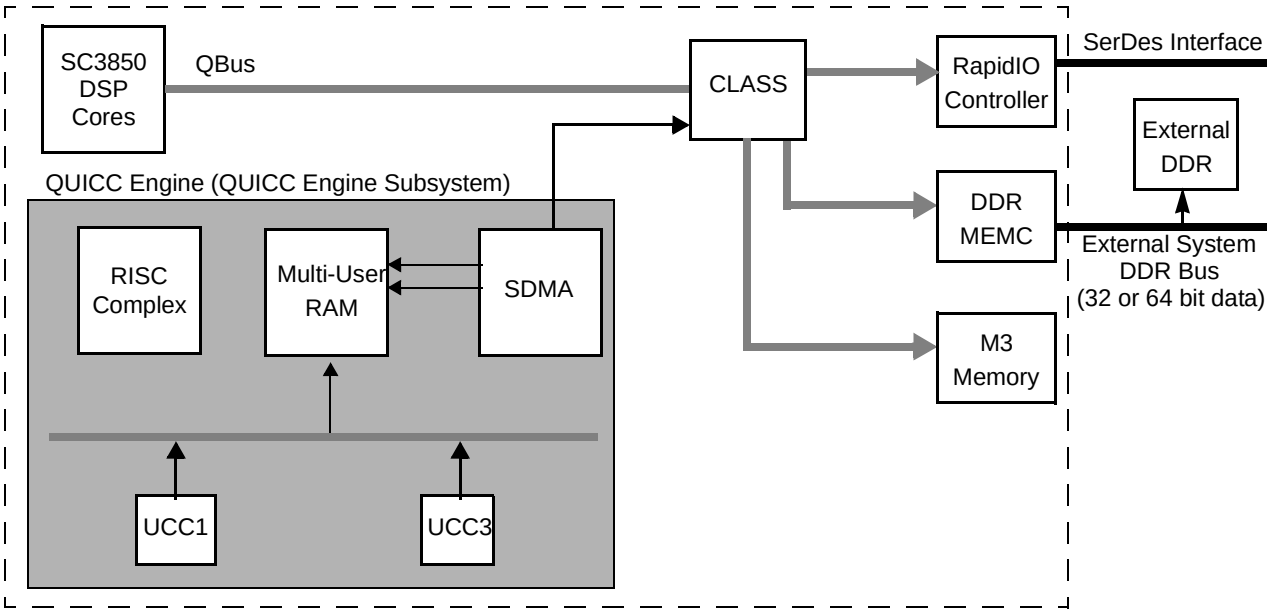


Figure 18-4. MSC8156E Data Paths

The CLASS is responsible for distribution of MBus transactions to the various possible targets (for example, RapidIO or DDR memory controller) based on the transaction address. See **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)* for details.

18.3.2 SDMA and Bus Error

If a bus error occurs on an SDMA access from the QUICC Engine subsystem, the following occurs:

1. The QUICC Engine subsystem generates a unique maskable interrupt in the SDMA status register (SDSR),
2. The DSP core uses an interrupt service routine (ISR) to read the SDSR to determine which bus generated the error.
3. System recovery depends on how you configure the SDMR[SBER_1] bit. One of the following occurs:
   - The QUICC Engine subsystem disables the peripheral or thread associated with the bus error and continues to operate as usual on all other peripherals (default). The recovery sequence in this mode is based on re-initialization of the peripheral associated with the error, or
   - The QUICC Engine subsystem stops all activity, and must be reset through the reset command to the QUICC Engine Command Register (CECR).

In general, it should be noted that the DSP core, depending on how it is programmed, may read the SDMA address register (SDTA) to determine the address at which the bus error occurred, and the SDMA SNUM register (SDTM) to determine which peripheral or thread was being serviced by the SDMA virtual channel. See **Table 18-3** for the list of SNUMs.

The SDTA and the SDTM registers store information related to accesses to the MBus. These two registers are not updated with the address and SNUM of subsequent transactions as long as the event bit in the SDSR is set.

18.3.2.1 Simple Recovery from Bus Error

The simplest recovery from a bus error is a QUICC Engine subsystem reset followed by an overall QUICC Engine subsystem re-initialization procedure, regardless of the peripheral that has actually caused the error. The reasoning is that for some applications, stopping one peripheral leads to a chain reaction that ultimately disrupts the correct interworking operation of the QUICC Engine subsystem. For debug purposes, it is valuable to observe continued operation, but a selective recovery does not provide any added value. This non-distinctive procedure also reduces the complexity of the recovery flow.

18.3.2.2 Selective Peripheral Recovery Procedure

Selective recovery can be a complex procedure due to the following:

- The Ethernet controllers are multi-thread controllers and SNUM to peripheral/controller translation requires maintaining association tables.
- Status for multiple bus errors is not maintained, so in theory there might be additional non-reported bus errors and the recovery will not be complete.

For these reason, a full reset and re-initialization are recommended.

18.3.3 SDMA and Reset

When the QUICC Engine subsystem is reset through the CECR[RST] bit, the SDMA continues to process outstanding transactions in its FIFOs although the data related to these transactions may be corrupted. During system reset ($\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$), all SDMA FIFOs are flushed and all outstanding transactions are stopped.

18.3.4 MBus Access

The SDMA requests the bus from the CLASS at two possible priority levels. When the SDMA is in normal state, it requests the bus at priority level programmed by the user in the SDMR[EBPR] bit field. When the SDMA is in emergency state, it requests the bus at the highest priority level that the CLASS supports; the QUICC Engine subsystem is assigned an arbitration weight through a field in GCR11 (see **Section 8.2.27**, *General Control Register 11 (GCR11)*, on page 8-42 for details). SDTR and SDHY program the threshold and hysteresis values that affect the conditions for SDMA normal and emergency states. It is possible to mask the emergency state priority requests globally in SDMR[EBMSK] and enforce the priority set in SDMR[EBPR] regardless of the SDMA state.

The SDMA asserts the highest priority request (emergency state) for any one of the following reasons:

- When one of the FIFOs in the QUICC Engine subsystem reaches an emergency state (too full on Rx or too empty in the middle of a frame during Tx)
- When the internal SDMA data buffers are filled beyond a certain level (programmed in SDTR/SDHY registers).
- When the internal SDMA command queue is filled beyond a certain level (programmed in SDTR/SDHY registers).

A priority request to the multi-user RAM is also asserted by the SDMA, if needed, for the same reasons. It is possible to mask the high priority request globally in SDMR[ERMSK].

18.3.5 SDMA Internal Resource

The SDMA requires temporary buffering for some data in the multi-user RAM. The base address for the SDMA temporary buffer is programmed in the SDEBCR. The base address must be aligned to a 4-KB boundary. The size of the multi-user RAM needed for this buffering is programmable via the STBSZ bit field in the SDMR. The size the temporary buffer that must be allocated ranges from a minimum of 512 bytes to maximum of 3 KB in the multi-user RAM. See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

18.4 Clocking

The QUICC Engine subsystem uses a programmable multiplexing system to route the various clocks used to transfer data through the external interfaces. Depending on the application and interface used, these signals can be supplied externally or taken from the internal programmable baud-rate generators. The following subsections describe the multiplexing unit and the internal baud-rate generators.

18.4.1 Multiplexer Logic

The QUICC Engine subsystem multiplexing logic routes clocks and connects the physical interfaces to the QUICC Engine subsystem peripherals, including the two UCCs. The multiplexer logic routes clocks to all the QUICC Engine subsystem peripherals from a bank of internal clocks (BRG[5–8]) and a bank of external clocks.

Physical signal connections are also configured using the clock route registers. However, because these signal lines are multiplexed with other functions at the I/O lines, you must make sure that the lines are also enabled through the GPIO registers and initial mode selection pins. **Figure 18-5** shows a block diagram of the QUICC Engine subsystem multiplexing logic.

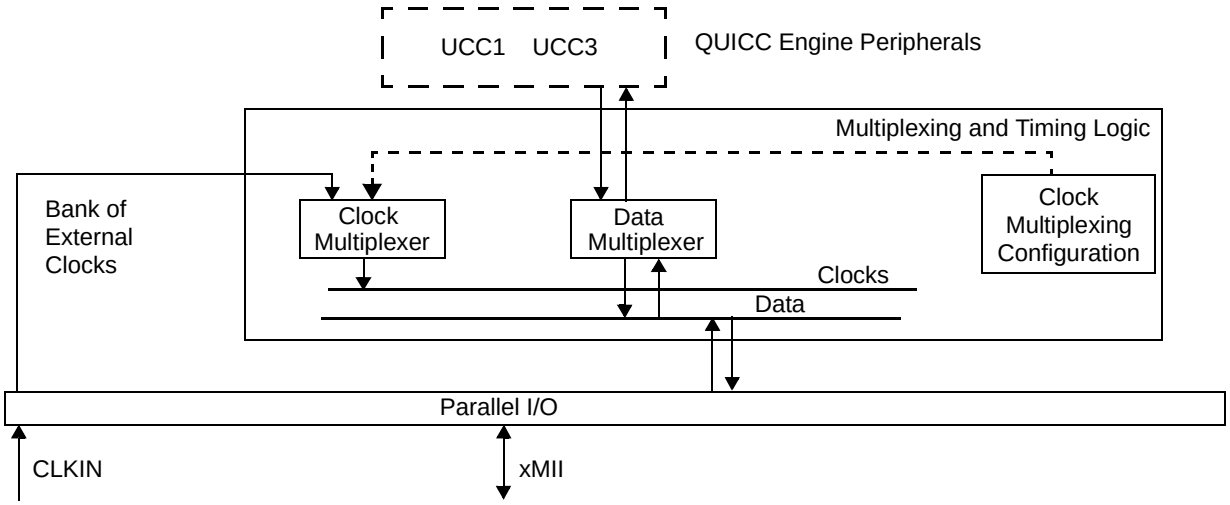


Figure 18-5. QUICC Engine Multiplexing Logic Block Diagram

The multiplexing logic is primarily used for clock assignment for UCC1 and UCC3. The clocks are derived from a bank of internal BRGs and external clock inputs as shown in **Figure 18-6**. The clock routing options are described in **Table 18-4** and **Table 18-5**. The bank of clocks selection logic applies an available clock to a peripheral that requires a clock. Because the peripheral is not directly connected to a specific clock source, peripherals can share the same clock. There are two main advantages to the bank-of-clocks approach. First, a peripheral is not forced to choose a serial device clock from a predefined input or BRG. Second, peripheral receivers and transmitters that need the same clock rate can share the same source. This configuration leaves additional signal lines for other functions and minimizes potential skew between multiple clock sources.

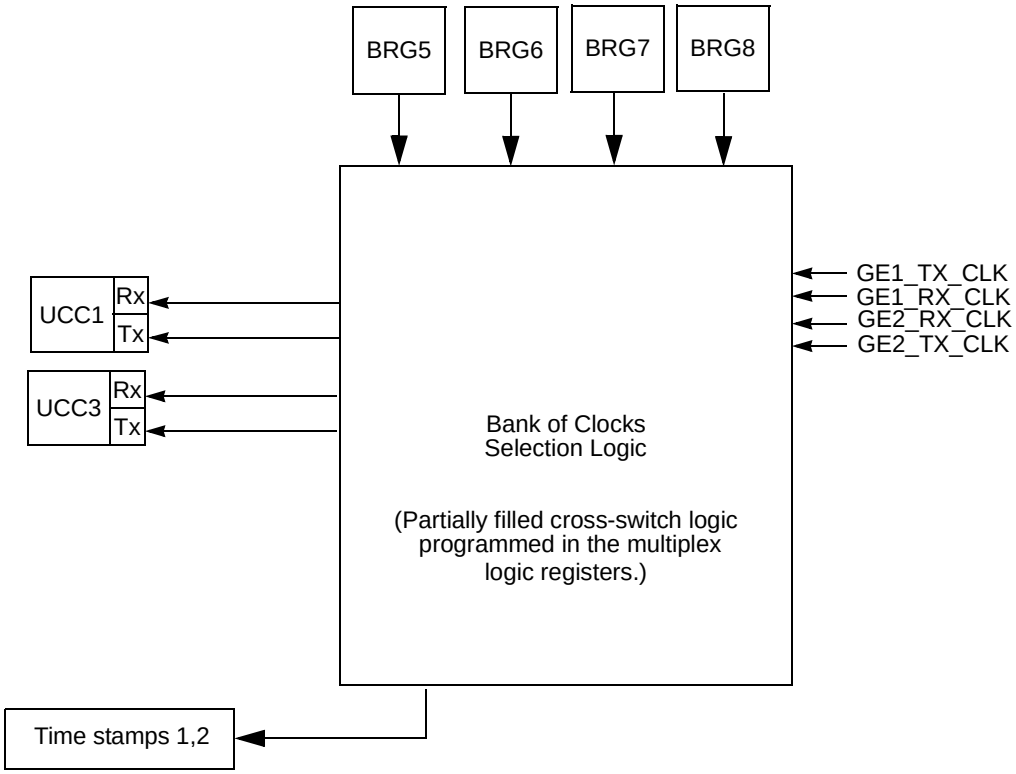


Figure 18-6. Bank of Clocks

Table 18-4. Clock Source Options Using External Clock Signals

Clock	External CLK			
	GE1_RX_CLK	GE1_TX_CLK	GE2_RX_CLK	GE2_TX_CLK
UCC1 Rx	V			
UCC1 Tx		V		
UCC3 Rx			V	
UCC3 Tx				V
Time Stamp 1			V	V
Time Stamp 2			V	V

Table 18-5. Clock Source Options - Internal Clock Generators

Clock	BRG Number			
	5	6	7	8
UCC1 Rx			V	
UCC1 Tx			V	
UCC3 Rx				V
UCC3 Tx				V

18.4.2 Baud-Rate Generators (BRGs)

The QUICC Engine subsystem contains four independent, identical baud-rate generators (BRGs) that can be used with the UCCs. The clocks produced by the BRGs are sent to the bank-of-clocks selection logic, where they can be routed to the controllers. Each BRG can be routed to one or more UCCs. **Figure 18-7** shows the block diagram for a BRG.

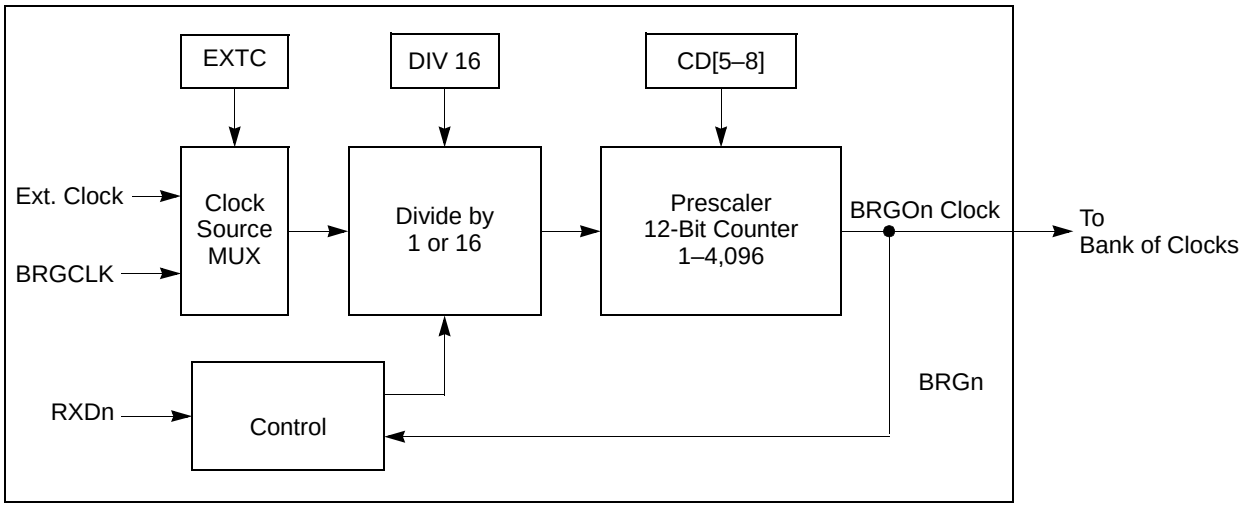


Figure 18-7. Baud-Rate Generator (BRG) Block Diagram

All BRGs can use BRGCLK as its source clock, or the external clock input selected by the value of BRGC*x*[EXTC]. The BRGCLK is an internal signal generated in the clock synthesizer. The external source option allows flexible baud-rate frequency generation, independent of the system frequency. Additionally, the external source option allows a single external frequency to be the source for more than one BRG. The external source signals are not synchronized internally before being used by the BRG. The BRG provides a divide-by-16 option (BRGC*x*[DIV16]) and a 12-bit prescaler (BRGC*x*[CD]) to divide the source clock frequency. The combined source-clock divide factor can be changed on-the-fly, except when changing to or from a CD value of 1, 2, or 3. For these values, disable the BRG and reset it before you program the new value. In addition, you should not make two changes within two source clock periods. If the BRG divides the clock by an even value, the transitions of BRGO*n* always occur on the rising edge of the source clock. If the divide factor is odd, the transitions alternate between the falling and rising edges of the source clock. The output of the BRG can be sent to the autobaud control block.

18.5 Interrupt Controller

The QUICC Engine subsystem interrupt controller sends general interrupts to the DSP cores in the MSC8156E device. The core must then initiate the correct interrupt service routine to handle the interrupt. Typically, this routine must read the interrupt status registers in the QUICC Engine subsystem to determine the appropriate action to take. For some specific interrupts, the MSC8156E uses five general configuration registers to expedite some interrupt responses:

- Four external request registers (CPE1R, CPE2R, CPE3R, and CPE4R) record events reported to the QUICC Engine subsystem directly by another MSC8156E peripheral (see **Section 8.2.16**, *QUICC Engine First External Request Multiplex Register (CPCE1R)*, on page 8-27 through **Section 8.2.19**, *QUICC Engine Fourth External Request Multiplex Register (CPCE4R)*, on page 8-30 for details).
- GIR1 includes two fields to specify whether an ECC error occurred for the QUICC Engine IRAM or DRAM (see **Section 8.2.21**, *General Interrupt Register 1 (GIR1)*, on page 8-32). These fields are maskable for each core individually using the corresponding fields in GIER1_x (see **Section 8.2.22**, *General Interrupt Enable Register 1 (GIER1_x)*, on page 8-34).

Note: See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details about configuring the QUICC Engine interrupt system.

18.6 UCCs

The QUICC Engine subsystem UCCs implement the Ethernet protocols. This section provides a general overview of the UCCs. For a detailed description of the feature set and the protocol-specific programming model, refer to the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM). The key features of the UCCs include:

- Supports 1000 Mbps, full-duplex Ethernet/IEEE 802.3x/VLAN through RGMII/SGMII.
- UCC clock can be derived from a internal clock or an external signal.
- Uses bursts to improve bus usage.
- Multibuffer data structure for received and transmitted data.
- Buffers and buffer descriptors (BDs) may reside anywhere in system memory.
- Programmable size-virtual FIFO buffers.
- Echo and local loopback modes for testing.

The UCC block diagram is shown in **Figure 18-8**.

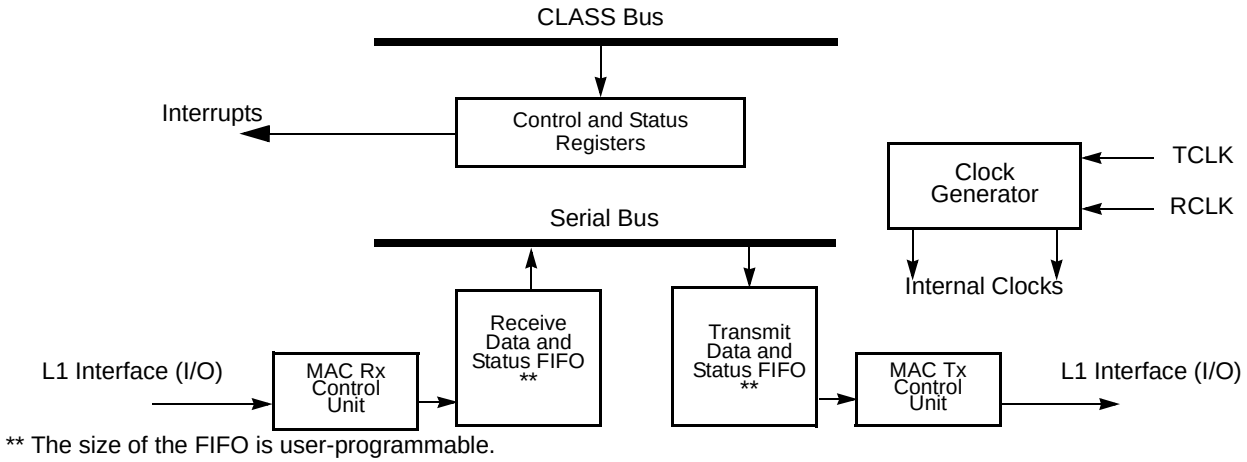


Figure 18-8. UCC Block Diagram

High-speed protocols require large FIFO depth. Sufficient FIFO size is important for increasing the QUICC Engine subsystem performance by eliminating overrun and underrun bottlenecks. Each protocol has an optimized FIFO size that depends not only on the bit rate, but also on other parameters such as packet or frame size. The UCC, when configured for high-speed protocols, extends the hardware FIFO into the QUICC Engine subsystem internal RAM. This extension is called virtual FIFO or VFIFO, because its size and location within the RAM are programmable according to the specific protocol. The FIFO as shown in **Figure 18-8** is constructed using a real hardware FIFO embedded in the UCC and its extension to a virtual FIFOs in the QUICC Engine subsystem RAM. This flexible FIFO structure enables the QUICC Engine subsystem to sustain wire speed frame or cell bursts by allocating larger FIFO memory when required. The size of the virtual FIFO is user-programmable. The actual size that is needed depends on a number of factors, especially the following:

- Maximum packet size
- Protocols running on the UCC
- Memory bus latency

18.7 Ethernet Controllers

The Ethernet interface is a widely-used local area network (LAN) that is based on the carrier-sense, multiple access, collision detect (CSMA/CD) approach. The **IEEE** 802.3 standard was developed to codify the requirements of such a system to insure interoperability between devices operating on the LAN. Because Ethernet and the **IEEE** protocols are similar and can coexist on the same LAN, this manual uses the generic term Ethernet unless otherwise noted. The MSC8156E uses two UCC Gigabit Ethernet Controllers (UECs) coordinated through the QUICC Engine subsystem. Each controller supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver. Supported interfaces include:

- 1000 Mbps RGMII interface
- 1000 Mbps SGMII interface.

The gigabit Ethernet modes were developed as individual company standards. As an alternative to the **IEEE** 802.3z (GMII) Ethernet standard that discuss general Ethernet interfacing, the RGMII Specification Reduced Pin-count Interface for Gigabit Ethernet Physical Layer Devices provides a method for Gigabit throughput while saving pins/board space. The RGMII Specification is managed by Broadcom, Marvell, and Hewlett-Packard, and is available on the Hewlett-Packard website at:

http://www.hp.com/rnd/pdfs/RGMIIv2_0_final_hp.pdf

To maintain Gigabit speed while reducing the data signals in half, the RGMI specification makes use of both the positive and negative edges of the clock. Because of this, meeting the RGMII specification requires careful attention to timing and delays.

A second protocol that uses an even lower pin count was developed by Cisco Systems. The Serial-GMH Specification defines a serial gigabit interface for Ethernet. The specification is available from the Cisco website at:

ftp://ftp-eng.cisco.com/smii/sgmii.pdf

The MSC8156E implements the SGMII through a SerDes interface managed by the HSSI (see **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem* for details.

Refer to the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for complete functional and programming details.

Figure 18-9 illustrates the block diagram of the UCC Ethernet controller.

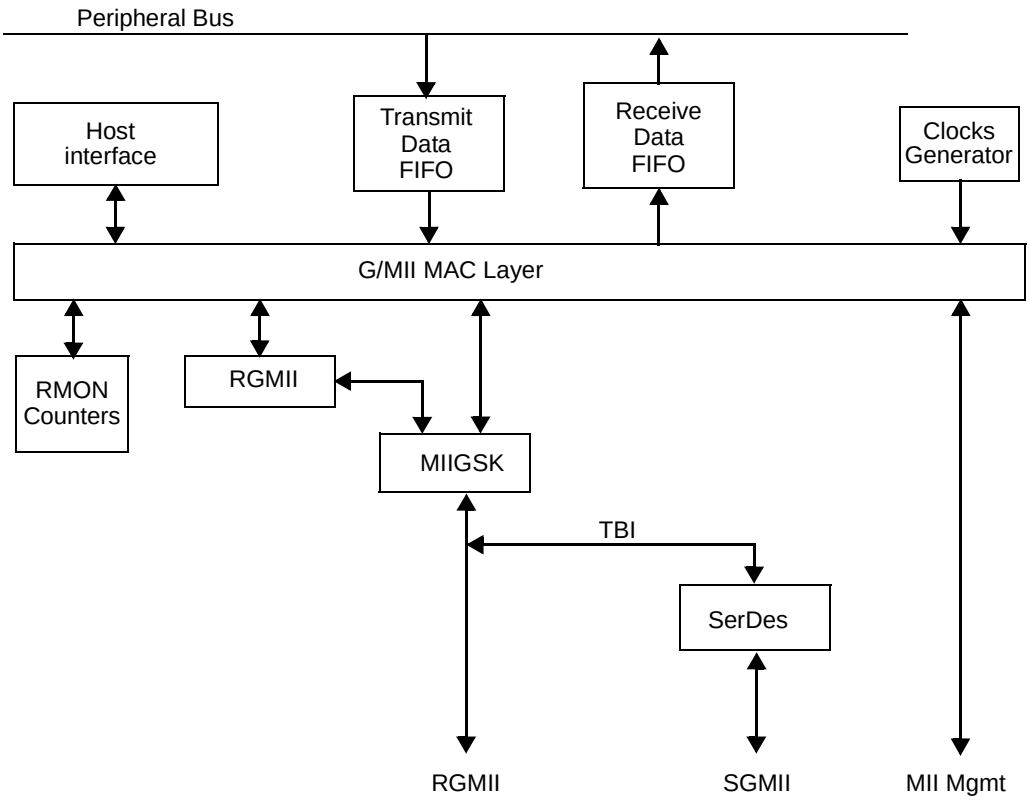


Figure 18-9. UCC Ethernet Controller Block Diagram

18.7.1 Operating Modes

Each Ethernet controller can be configured to use RGMII or SGMII mode. Two fields in the QUICC Engine Control Register allow programming of each controller independently (see **Section 8.2.8**, *QUICC Engine Control Register (QECR)*, on page 8-16 for details).

18.7.1.1 RGMII Mode

The RGMII is intended to be an alternative to the **IEEE** 802.3u MII, the **IEEE** 802.3z GMII, and TBI standards. The principle objective is to reduce the number of pins required to interconnect the MAC and the PHY from a maximum of 28 pins (TBI) to 12 pins in a cost effective and technology independent manner. In order to accomplish this objective, the data paths and all associated control signals are reduced and control signals are multiplexed together and both edges of the clock are used. For Gigabit operation, the clocks operate at 125 MHz.

18.7.1.2 SGMII Mode

The Serial Gigabit Media Independent Interface (SGMII) is designed to satisfy the following requirements:

- Convey network data and port speed between a 1000 PHY and a MAC with significantly less signal pins than required for GMII.
- Operate in full duplex.

In the MSC8156E, SGMII is implemented used the SerDes interface.

Note: The internal ten-bit interface (TBI) circuitry is used to serialize/deserialize the Ethernet frame data. The TBI must be configured to perform this function. See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for programming details.

18.7.2 Ethernet Physical Interfaces

You can program the physical interfaces by configuring the PSMR and MACCFG2 registers. See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

Table 18-6 lists the external signal properties of the shared Management Data lines.**Chapter 3**, *External Signals* describes the assignment of the signals to the device pins.

Table 18-6. Management Data Signal Properties

Name	Function	I/O
MDC	**Management Data Clock for all Ethernet interfaces** The MDIO signal clock reference (2.5 MHz clock).	Output
MDIO	**Management Data Input/Output for all Ethernet interfaces** Transfers control signals between the PHY layer and the manger entity.	Input/ Output

The following subsections give details on the RGMII and SGMII signals.

18.7.2.1 Reduced Gigabit Media-Independent Interface (RGMII) Signals

The RGMII data paths and all associated control signals are reduced, control signals are multiplexed, and both edges of the clock are used. **Figure 18-10** shows the RGMII connections between the Ethernet controller and a PHY.

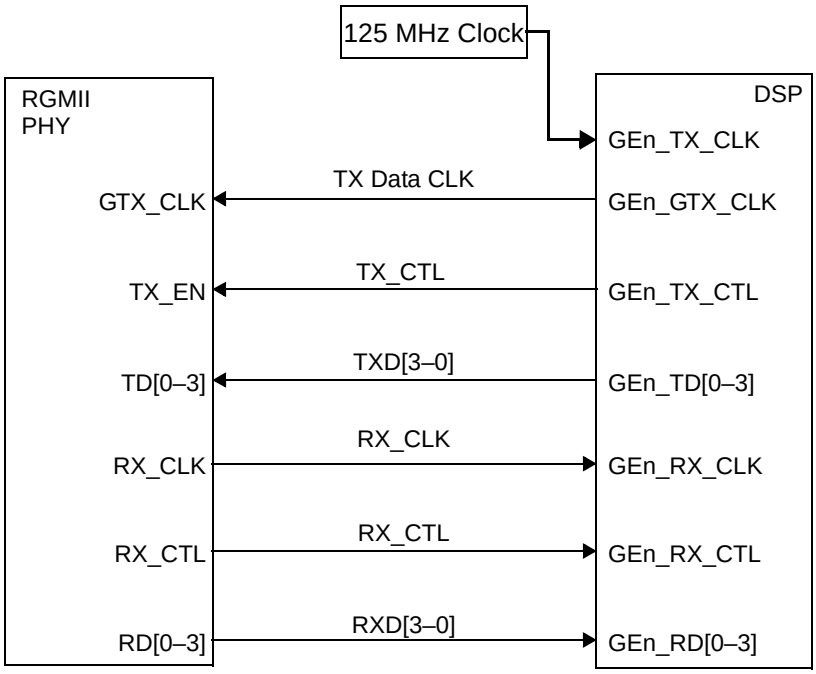


Figure 18-10. RGMII MAC-PHY Interface

18.7.2.1.1 RGMII Signals

Table 18-7 lists the RGMII signals.

Table 18-7. RGMII Signals

Consortium Name	I/O	Size	Function	Reference Clock
GTX_CLK	O	1	**Transmit Reference Clock** 125 MHz	—
TX_CLK	I	1	**Transmit Clock**	GTX_CLK
TX_CTL	O	1	**Transmit Control** TX_EN on clock positive edge. TX_ER on clock negative edge.	TXC
Tx Data	O	4	**Transmit Data** TXD[0–3] on clock positive edge. TXD[4–7] on clock negative edge.	TXC

Table 18-7. RGMII Signals (Continued)

Consortium Name	I/O	Size	Function	Reference Clock
RX_CTL	I	1	**Receive Control** RX_DV on clock positive edge. RX_ER on clock negative edge.	RXC
Rx Data	I	4	**Receive Data** RXD[0–3] on clock posedge RXD[4–7] on clock negedge	RXC
MDIO	I/O	1	**Management Data I/O** Transfers control signals between the PHY layer and the manager entity.	MDC
MDC	O	1	**Management Data Clock** The MDIO signal clock reference (2.5 MHz clock).	—
RX_CLK	I	1	**Continuous Receive Reference Clock** 125 MHz	—

18.7.2.1.2 RGMII Signal Configuration

The RGMII signals are multiplexed with the TDM signals (see **Chapter 3**, *External Signals*). Ethernet controller 1 is multiplexed with TDM2 and TDM3 signals and Ethernet controller 2 is multiplexed with TDM0 and TDM1 signals. Selection is done at reset by the value of the GE1 and GE2 fields in the Reset Configuration Word High (see **Section 5.3.2**, *Reset Configuration Word High Register (RCWHR)*, on page 5-19 for details). If the value of field is 0 (low), the TDM signals are selected; if the value of the field is 1 (high), the Ethernet signals are selected. In addition, the appropriate field in the QUICC Engine Control Register (ENET_SGMII_MODE1 for GE2 or ENET_SGMII_MODE0 for GE1) must be configured for RGMII (the field must be 0) (see **Section 8.2.8**, *QUICC Engine Control Register (QECR)*, on page 8-16 for details).

Note: The MSC8156E allows adjustment of the transmission delays for the RGMII signal lines using GCR4 (see **Section 8.2.12**, *General Control Register 4 (GCR4)*, on page 8-20 for register details). Recommended settings are listed in the MSC8156E data sheet. Guidelines for adjusting these numbers in individual designs is provided in *Using GCR4 to Adjust Ethernet Timing in MSC8144 DSPs* (AN3811), available at www.freescale.com. The adjustment recommendations are applicable for the MSC8156E DSP.

18.7.2.2 Serial Gigabit Media-Independent Interface (SGMII) Signals

The SGMII is an alternative to the RGMII interface that further reduces the number of pins required to interconnect the MAC and PHY by using a SerDes 4 interface. It does not support auto-negotiation. The Ethernet controller SGMII interface uses the UEC ten-bit interface (TBI) connection internally, which connects to the SerDes block that serializes the transmitted data and deserializes the received data to/from the SGMII interface.

18.7.2.2.1 SGMII Signals

The SGMII physical interface transmits and receives data using two data signals and a clock signals to convey frame data and link rate information between a 1000 Mbps PHY and an Ethernet MAC. The data signals operate at 1.25 Gbaud. Due to the speed of operation, each of these signals (including the clock signal) is realized as a differential pair thus providing signal integrity while minimizing system noise. Therefore, each data and clock signal path uses two physical signal lines (the differential pair). **Table 18-8** lists the SGMII signals.

Table 18-8. SGMII Signals

Signal Name	I/O	Size	Function	Reference Clock
SRIO_REF_CLK $\overline{\text{SRIO\_REF\_CLK}}$	I I	2	**Reference Clock** 125 MHz differential pair.	—
SG1_TX $\overline{\text{SG1\_TX}}$	O O	2	**Transmit Data 1** Differential pair for Ethernet 1 controller.	$\overline{\text{SRIO\_REF\_CLK}}$ SRIO_REF_CLK
SG2_TX $\overline{\text{SG2\_TX}}$	O O	2	**Transmit Data 2** Differential pair for Ethernet 2 controller.	$\overline{\text{SRIO\_REF\_CLK}}$ SRIO_REF_CLK
SG1_RX $\overline{\text{SG1\_RX}}$	I I	2	**Receive Data 1** Differential pair for Ethernet 1 controller.	$\overline{\text{SRIO\_REF\_CLK}}$ SRIO_REF_CLK
SG2_RX $\overline{\text{SG2\_RX}}$	I I	2	**Receive Data 2** Differential pair for Ethernet 2 controller.	$\overline{\text{SRIO\_REF\_CLK}}$ SRIO_REF_CLK
MDIO	I/O	1	**Management Data I/O** Transfers control signals between the PHY layer and the manager entity.	MDC
MDC	I	1	**Management Data Clock** The MDIO signal clock reference (25 MHz clock).	—

18.7.2.2.2 SGMII Signal Configuration

The SGMII signals are multiplexed with the serial RapidIO and PCI Express signals (see **Chapter 3**, *External Signals* and **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*). Either signal can be routed through SerDes port 1 or 2, depending on the selected multiplex configuration. Selection is done at reset by the value of the S2P and S1P fields in the Reset Configuration Word Low (see **Section 5.3.1**, *Reset Configuration Word Low Register (RCWLR)*, on page 5-17 for details). The values of these fields determine which lanes of the SerDes ports are assigned to the SGMII signals. In addition, the appropriate field in the QUICC Engine Control Register (ENET_SGMII_MODE1 for GE2 or ENET_SGMII_MODE0 for GE1) must be configured for SGMII (the field must be 1) (see **Section 8.2.8**, *QUICC Engine Control Register (QECR)*, on page 8-16 for details).

18.7.3 Controlling PHY Links (Management Interface)

The support for MII Ethernet Management can be done by the SPI or by one UCC that can be selected using CMXGCR[SMI]. Control and status to and from the PHY is provided via the two-wire MII management interface described in the **IEEE** 802.3u standard. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) exercise this interface between a host processor and one or more PHY devices.

The UEC MII registers support continuous read cycles (called a scan cycle); even through scan cycles are not explicitly defined in the standard. If requested (by setting MIIMCOM[scan cycle]), the controller performs repetitive read cycles of the PHY status register, for example. This allows you to monitor link characteristics more efficiently. The different fields in the MII management indicator register (scan, not valid, and busy) indicate availability of each read of the scan cycle to the host via MIIMSTAT[PHY scan] bit field.

The length of the MII management interface preamble can also be modified through the MII registers. After establishing that a PHY supports preamble suppression, the host may configure the UEC to suppress the preamble. When enabled, the length of MII management frames are reduced from 64 to 32 clocks. This effectively doubles the efficiency of the interface.

18.7.4 Ethernet Controller Initialization

After the Ethernet Controller completes the reset sequence, software must initialize certain UEC registers and the required parameters in the parameter RAM. Based on system requirements, other optional registers and parameters can also be initialized at the same time.

Table 18-9 lists the minimum steps required for register and parameter initialization

Table 18-9. Minimum Register Initialization

Initialization Step	Registers
Configure the UCC to Fast protocols	URMODE,UTMODE
Set the Tx Global Parameter RAM	
Set the Rx Global Parameter RAM	
Set CMXUCR1	Select UCC1, UCC3 RxClk and TxClk
Initialize the MAC Station address	MACSTNADDR1 and MACSTNADDR2
Initialize the Media media access control configuration register and the UCC protocol specific mode register	This MACCFG2 together with UPSMR register adjust frame length and preamble length, specifies various CRC/pad combinations, specifies Full/Half Duplex and operating mode
Initialize the Fast Protocol Fifo Configuration registers	URFB, URFET, URFS, URFSET, UTFB, UTFS, UTFET, UTFTT, URTRY
UCC event register, Fast UCCE	Initialize interrupts to prepare for interrupt events
UCC mask register, Fast UCCM	Initialize the interrupt mask to prepare for interrupt events
Activate The Ethernet Controller	

Table 18-9. Minimum Register Initialization (Continued)

Initialization Step	Registers
Initialize the InitEnet parameter	CECDR
Initialize the Tx and Rx parameters of UCC1 ethernet.	CECR
Enable the Ethernet Controller MAC TX and RX	MACCFG1
Note: See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for register addressing, structure, and programming details.	

After initializing the registers, you must complete the following steps to bring the Ethernet Controller into a functional state:

1. To transmit Ethernet frames, build the TxBDs in memory, link them together as a ring, and point to the ring. A minimum of two TxBDs per ring is required.
2. To receive Ethernet frames, link the RxBDs together as a ring and point the corresponding registers to them. Both transmit and receive can be gracefully stopped after transmission and reception begins.

For SGMII mode, in addition to the minimum initialization steps, based on your system requirements, you must configure the QUICC Engine Control Register (QECR) to work in SGMII mode. See **Section 8.2.8, QUICC Engine Control Register (QECR)** on page 8-16 for details. You must also configure the TBI MII registers. See the *QUICC Engine Block Reference Manual with Protocol Interworking* (QEIWRM) for details.

18.8 Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the exchange of data with other devices containing an SPI. The SPI also communicates with peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices. The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock, and slave select). The SPI block 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 SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is sent and received simultaneously. The SPI receiver and transmitter are double-buffered, as shown in **Figure 18-11**, giving an effective FIFO size (latency) of 2 characters. When the SPI is disabled in the SPI mode register (SPMODE[EN] = 0), it consumes little power.

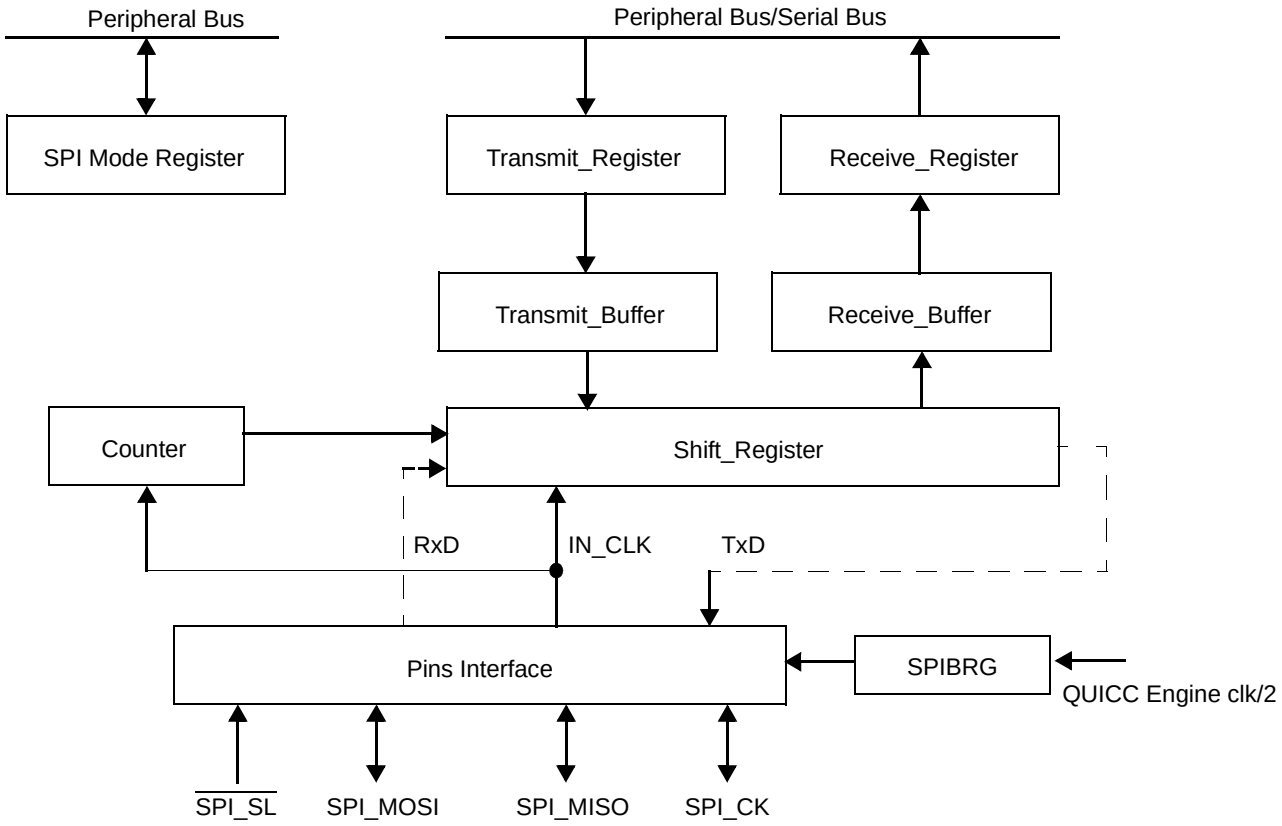


Figure 18-11. SPI Block Diagram

18.8.1 SPI Operating Modes

The SPI can be programmed to work in a single- or multiple-master environment. This section describes the SPI master and slave operation in a single-master configuration and then discusses the multi-master environment. The following sections present a summary of the main modes of operation which the SPI supports.

18.8.1.1 SPI as a Master Device

In master mode, the SPI sends a message to the slave peripheral, which sends back a simultaneous reply. A single-master device with multiple slaves can use general-purpose parallel I/O signals to selectively enable slaves, as shown in **Figure 18-12**. To eliminate the multi-master error in a single-master environment, the master $\overline{\text{SPI\_SL}}$ input can be forced inactive by selecting $\overline{\text{SPI\_SL}}$ for general-purpose I/O.

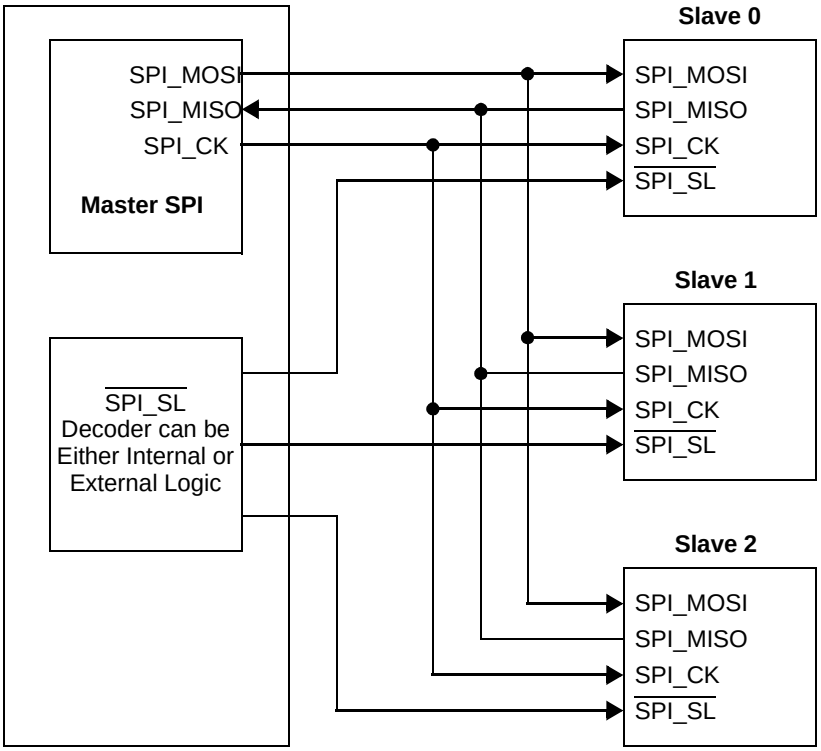


Figure 18-12. Single-Master/Multi-Slave Configuration

To start exchanging data, the QUICC Engine subsystem writes the data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The QUICC Engine subsystem then sets SPCOM[STR] in the SPI command register to start sending data, which starts once the SDMA channel loads the Tx FIFO with data.

The SPI then generates programmable clock pulses on SPI_CK for each character and simultaneously shifts Tx data out on SPI_MOSI and Rx data in on SPI_MISO. Received data is written into a Rx buffer using the next available RxBD. The SPI keeps sending and receiving characters until the whole buffer is sent or an error occurs. The QUICC Engine subsystem then clears TxBD[R] and RxBD[E] and issues a maskable interrupt to the interrupt controller.

When multiple TxBDs are ready, TxBD[L] determines whether the SPI keeps transmitting without SPCOM[STR] being set again. If the current TxBD[L] is cleared, the next TxBD is processed after data from the current buffer is sent. Typically, there is no delay on SPI_MOSI between buffers. If the current TxBD[L] is set, sending stops after the current buffer is sent. In addition, the RxBD is closed after transmission stops, even if the Rx buffer is not full; therefore, Rx buffers need not be the same length as Tx buffers.

18.8.1.2 SPI as a Slave Device

In slave mode, the SPI receives messages from an SPI master and sends a simultaneous reply. The slave's $\overline{\text{SPI\_SL}}$ must be asserted before Rx clocks are recognized; once $\overline{\text{SPI\_SL}}$ is asserted, SPI_CK becomes an input from the master to the slave. SPI_CK can be any frequency from DC to QUICC Engine clk/4.

To prepare for data transfers, the slave's core processor writes data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core processor then sets SPCOM[STR] to activate the SPI. Once $\overline{\text{SPI\_SL}}$ is asserted, the slave shifts data out from SPI_MISO and in through SPI_MOSI. A maskable interrupt is issued when a full buffer finishes receiving and sending or after an error. The SPI uses successive RxBDs in the table to continue reception until it runs out of Rx buffers or $\overline{\text{SPI\_SL}}$ is deasserted.

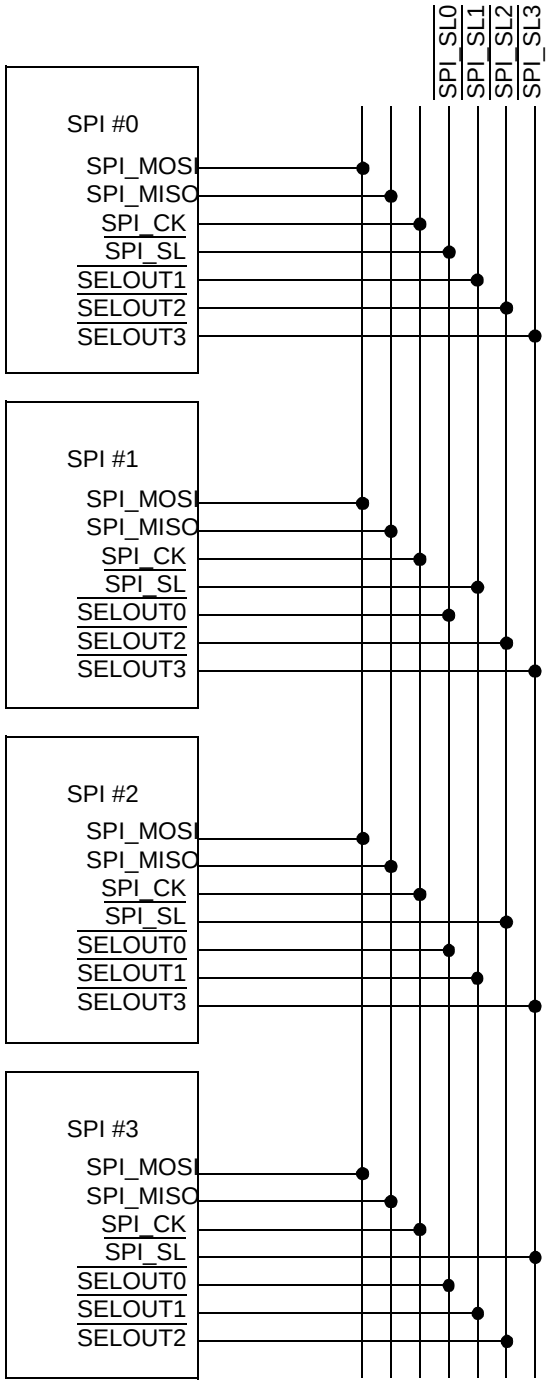
Transmission continues until no more data is available or $\overline{\text{SPI\_SL}}$ is deasserted. If it is deasserted before all data is sent, it stops but the TxBD stays open. Transmission continues once $\overline{\text{SPI\_SL}}$ is reasserted and SPI_CK begins toggling. After the characters in the buffer are sent, the SPI sends ones as long as $\overline{\text{SPI\_SL}}$ remains asserted.

Note: When enabling the SPI or changing parameters in SPI Mode Register (like CP,CI), $\overline{\text{SPI\_SL}}$ must remain deasserted for at least 2 QUICC Engine clk/2 clocks afterwards. Also if $\overline{\text{SPI\_SL}}$ is deasserted between transfers, its deassertion time should be at least 2 QUICC Engine clk/2 clocks.

18.8.2 SPI in Multi-Master Operation

The SPI can operate in a multi-master environment in which SPI devices are connected to the same bus. In this configuration, the SPI_MOSI, SPI_MISO, and SPI_CK signals of all SPIs are shared; the $\overline{\text{SPI\_SL}}$ inputs are connected separately, as shown in **Figure 18-13**. Only one SPI device at a time can act as a master—all others must be slaves. When an SPI is configured as a master and its $\overline{\text{SPI\_SL}}$ input is asserted, a multi master error occurs because more than one SPI device is a bus master. The SPI sets SPIE[MME] in the SPI event register and a maskable interrupt is issued to the QUICC Engine subsystem. It also disables SPI operation and the output drivers of SPI signals. The core processor must clear SPMODE[EN] before the SPI is used again. After correcting the problems, clear SPIE[MME] and re-enable the SPI.

The maximum sustained data rate that the SPI supports is QUICC Engine clk/50. However, the SPI can transfer a single character at much higher rates—QUICC Engine clk/8 in master mode and QUICC Engine clk/4 in slave mode. Gaps should be inserted between multiple characters to keep from exceeding the maximum sustained data rate.



Notes:
- All signals are open-drain
- For a multi-master QUICC Engine subsystem with more than two masters, SPI_SL and SPIE[MME] will not detect all possible conflicts.
- It is the responsibility of the software to arbitrate for the SPI bus (with token passing, for example).
- SPI_SLx signals are implemented in the software with general-purpose I/O signals.

Figure 18-13. Multimaster Configuration

18.8.3 External Signal Configuration

The SPI supports a four-wire interface—transmit, receive, clock, and slave select. See **Chapter Figure 3-1.**, *MSC8156E External Signals* for detailed signal descriptions. After reset, the signals are assigned as GPIO17 to GPIO20. They must be configured as SPI signals by writing the correct values to the GPIO configuration registers. Refer to **Table 3-9 *SPI Signals*** on page 3-20 and **Chapter 22**, *GPIO* for programming information.

The SPI can be configured as a slave or as a master in single- or multiple-master environments. The master SPI generates the transfer clock SPI_CK, using the SPI baud rate generator (BRG). The SPI BRG input is QUICC Engine clk /2. The selection as slave or master determines the signal direction (input or output) to select when configuring the signals using the GPIO configuration registers.

SPI_CK is a gated clock, active only during data transfers. Four combinations of SPI_CK phase and polarity can be configured by using SPMODE[CI, CP]. SPI signals can also be configured as open-drain to support a multimaster configuration in which a shared SPI signal is driven by the processor or an external SPI device.

The SPI 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. The dual functionality of these signals allows the SPIs in a multimaster environment to communicate with one another using a common hardware configuration.

- When the SPI is a master, SPI_CK is the clock output signal that shifts received data in from SPI_MISO and transmitted data out to SPI_MOSI. SPI masters must output a slave select signal to enable SPI slave devices by using a separate general-purpose I/O signal. Assertion of $\overline{\text{SPI\_SL}}$ while it is a master causes an error.
- When the SPI is a slave, SPI_CK is the clock input that shifts received data in from SPI_MOSI and transmitted data out through SPI_MISO. $\overline{\text{SPI\_SL}}$ is the input enable to the SPI slave. In a multi-master environment, $\overline{\text{SPI\_SL}}$ (always an input) is also used to detect an error when more than one master is operating.

18.8.4 SPI Transmission and Reception Process

Because the SPI is a character-oriented communication unit, the core processor must pack and unpack the receive/transmit frames. A frame consists of all of the characters transmitted or received during a completed SPI transmission session, from the first character written to the internal SPI transmit data register (SPITD) to the last character transmitted following the setting of the LST bit in the SPI command (SPCOM) register.

The core processor receives data by reading the SPI receive data register (SPIRD) when the SPI event register (SPIE) not-empty bit (SPIE[NE]) is set. The core processor transmits data by writing it into the SPITD. When the next character to transmit is the final one in the current

frame, the core processor sets the last (SPCOM[LST]) bit and then writes the final character to SPITD. The SPI sets the not-full (SPIE[NF]) bit whenever its transmit FIFO is not full. It clears the bit when the last character is written to SPITD and resets it after sending the last data.

The SPI-core processor handshake protocol can use a polling or interrupt mechanism. When using polling, the core processor reads the SPIE at a predefined frequency and acts according to the value of the SPIE bits. The polling frequency depends on the SPI serial channel frequency. When using the interrupt mechanism, setting either SPIE[NF] or SPIE[NE] causes an interrupt to the core processor. The core processor then reads the SPIE and acts accordingly. There are three basic modes of operation for transmitting and receiving: master, slave, and multimaster.

18.9 Programming Model

This section provides a summary list of the MSC8156E QUICC Engine subsystem, Ethernet controller, and SPI registers with their offsets.

Note: The QUICC Engine registers use a base address of 0xFEE00000.

Table 18-10. MSC8156E QUICC Engine Register Summary

Register Name	Acronym	Offset
IRAM Registers		
IRAM Address Register	IADD	0x0000
IRAM Data Register	IDATA.	0x0004
Interrupt Controller Registers		
QUICC Engine System Interrupt Configuration Register	CICR.	0x0080
QUICC Engine Interrupt Vector Register	CIVEC.	0x0084
QUICC Engine RISC Interrupt Pending Register	CRIPNR.	0x0088
QUICC Engine System Interrupt Pending Register	CIPNR.	0x008C
QUICC Engine Interrupt Priority Register—XCC Peripherals	CIPXCC.	0x0090
QUICC Engine Interrupt Priority Register—WCC Peripherals	CIPWCC.	0x0098
QUICC Engine Interrupt Priority Register—ZCC Peripherals	CIPZCC.	0x009C
QUICC Engine System Interrupt Mask Register	CIMR.	0x00A0
QUICC Engine RISC Interrupt Mask Register	CRIMR.	0x00A4
QUICC Engine System Interrupt Control Register	CICNR.	0x00A8
QUICC Engine Interrupt Priority Register for RISC Tasks A	CIPRTA.	0x00B0
QUICC Engine System RISC Interrupt Control Register	CRICR.	0x00BC
QUICC Engine High System Interrupt Vector Register	CHIVEC.	0x00E0
QUICC Engine System		
QUICC Engine Command Register	CECR	0x0100
QUICC Engine Command Data Register	CECDR	0x0108

Table 18-10. MSC8156E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
QUICC Engine Time-Stamp Control Register	CETSCR	0x011C
QUICC Engine Virtual Tasks Event Register	CEVTER	0x0130
QUICC Engine Virtual Tasks Mask Register	CEVTMR	0x0134
QUICC Engine RAM Control Register	CERCR	0x0138
QUICC Engine Microcode Revision Number	CEURNR	0x01B8
QUICC Engine Multiplexer Registers		
CMX General Clock Route Register	CMXGCR	0x0400
UCC Clock Route Register 1	CMXUCR1.	0x0410
SPI registers		
SPI Mode Register	SPMODE.	0x04E0
SPI Event Register	SPIE.	0x04E4
SPI Mask Register	SPIM.	0x04E8
SPI Command Register	SPCOM.	0x04EC
Baud Rate Generators		
Baud-Rate Generator Configuration Registers 5	BRGCR5	0x0650
Baud-Rate Generator Configuration Registers 6	BRGCR6	0x0654
Baud-Rate Generator Configuration Registers 7	BRGCR7	0x0658
Baud-Rate Generator Configuration Registers 8	BRGCR8	0x065C
UCC Registers		
UCC1 General Mode Register	GUMR1.	0x2000
UCC1 Protocol-Specific Mode Register	UPSMR1.	0x2004
UCC1 Transmit On Demand Register	UTODR1.	0x2008
UCC1 Event Register	UCCE1.	0x2010
UCC1 Mask Register	UCCM1.	0x2014
UCC1 Ethernet Transmitter Status Register	UCCS1	0x2018
UCC1 Receive FIFO Base	URFB1.	0x2020
UCC1 Receive FIFO Size	URFS1.	0x2024
UCC1 Receive FIFO Emergency Threshold	URFET1.	0x2028
UCC1 Receive FIFO Special Emergency Threshold	URFSET1.	0x202A
UCC1 Transmit FIFO Base	UTFB1.	0x202C
UCC1 Transmit FIFO Size	UTFS1	0x2030
UCC1 Transmit FIFO Emergency Threshold	UTFET1.	0x2034
UCC1 Transmit FIFO Transmit Threshold	UTFTT1.	0x2038
UCC1 Transmit Polling Timer	UFPT1	0x203C
UCC1 Retry Counter	URTRY1.	0x2040
UCC1 General Extended Mode Register	GUEMR1.	0x2090

Table 18-10. MSC8156E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
Ethernet 1 MAC Configuration 1 Register	E1MACCFG1	0x2100
Ethernet 1 MAC Configuration 2 Register	E1MACCFG2	0x2104
Ethernet 1 Interframe Gap Register	E1PGFG	0x2108
Ethernet 1 Half Duplex Register	HAFDUP1	0x210C
Ethernet 1 MII Management Configuration Register	MIIMCFG1	0x2120
Ethernet 1 MII Management Command Register	MIIMCOM1	0x2124
Ethernet 1 MII Management Address Register	MIIMADD1	0x2128
Ethernet 1 MII Management Control Register	MIIMCON1	0x212C
Ethernet 1 MII Management Status Register	MIIMSTAT1	0x2130
Ethernet 1 MII Management Indicator Register	MIIMIND1	0x2134
Ethernet 1 Interface Status Register	IFSTAT1	0x213C
Ethernet 1 Station Address Part 1 Register	E1MACSTNADDR1	0x2140
Ethernet 1 Station Address Part 2 Register	E1MACSTNADDR2	0x2144
Ethernet 1 MAC Parameter Register	UEMPR1	0x2150
Ethernet 1 Ten-Bit Interface Physical Address Register	UTBIPAR1	0x2154
Ethernet 1 Statistics Control Register	UESCR1	0x2158
Ethernet 1 Tx 64-byte Frames	E1TX64	0x2180
Ethernet 1 Tx 65- to 127-byte Frames	E1TX127	0x2184
Ethernet 1 Tx 128- to 255-byte Frames	E1TX255	0x2188
Ethernet 1 Rx 64-byte Frames	E1RX64	0x218C
Ethernet 1 Rx 65- to 127-byte Frames	E1RX127	0x2190
Ethernet 1 Rx 128- to 255-byte Frames	E1RX255	0x2194
Ethernet 1 Octet Transmitted OK	E1TXOK	0x2198
Ethernet 1 Tx Pause Frames	E1TXCF	0x219C
Ethernet 1 Multicast Frame Transmitted OK	E1TMCA	0x21A0
Ethernet 1 Broadcast Frames Transmitted OK	E1TBCA	0x21A4
Ethernet 1 Number of Frames Received OK	E1RXFOK	0x21A8
Ethernet 1 Rx Octets OK	E1RBYT	0x21AC
Ethernet 1 Rx Octets	E1RXBOK	0x21B0
Ethernet 1 Multicast Frame Received OK	E1RMCA	0x21B4
Ethernet 1 Broadcast Frames Received OK	E1RBCA	0x21B8
Ethernet 1 Statistic Counters Carry Register	E1SCAR	0x21BC
Ethernet 1 Statistic Counters Carry Mask Register	E1SCAM	0x21C0
UCC3 General Mode Register	GUMR3.	0x2200
UCC3 Protocol-Specific Mode Register	UPSMR3.	0x2204
UCC3 Transmit On Demand Register	UTODR3.	0x2208
UCC3 Event Register	UCCE3.	0x2210

Table 18-10. MSC8156E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
UCC3 Mask Register	UCCM3.	0x2214
UCC3 Ethernet Transmitter Status Register	UCCS3	0x2218
UCC3 Receive FIFO Base	URFB3.	0x2220
UCC3 Receive FIFO Size	URFS3.	0x2224
UCC3 Receive FIFO Emergency Threshold	URFET3	0x2228
UCC3 Receive FIFO Special Emergency Threshold	URFSET3	0x222A
UCC3 Transmit FIFO Base	UTFB3.	0x222C
UCC3 Transmit FIFO Size	UTFS3	0x2230
UCC3 Transmit FIFO Emergency Threshold	UTFET3	0x2234
UCC3 Transmit FIFO Transmit Threshold	UTFTT3.	0x2238
UCC3 Transmit Polling Timer	UFPT3	0x223C
UCC3 Retry Counter	URTRY3.	0x2240
UCC3 General Extended Mode Register	GUEMR3	0x2290
Ethernet 2 MAC Configuration 1 Register	E2MACCFG1	0x2300
Ethernet 2 MAC Configuration 2 Register	E2MACCFG2	0x2304
Ethernet 2 Interframe Gap Register	E2PGFG	0x2308
Ethernet 2 Half Duplex Register	HAFDUP2	0x230C
Ethernet 2 MII Management Configuration Register	MIIMCFG2	0x2320
Ethernet 2 MII Management Command Register	MIIMCOM2	0x2324
Ethernet 2 MII Management Address Register	MIIMADD2	0x2328
Ethernet 2 MII Management Control Register	MIIMCON2	0x232C
Ethernet 2 MII Management Status Register	MIIMSTAT2	0x2330
Ethernet 2 MII Management Indicator Register	MIIMIND2	0x2334
Ethernet 2 Interface Status Register	IFSTAT2	0x233C
Ethernet 2 Station Address Part 1 Register	E2MACSTNADDR1	0x2340
Ethernet 2 Station Address Part 2 Register	E2MACSTNADDR2	0x2344
Ethernet 2 MAC Parameter Register	UEMPR2	0x2350
Ethernet 2 Ten-Bit Interface Physical Address Register	UTBIPAR2	0x2354
Ethernet 2 Statistics Control Register	UESCR2	0x2358
Ethernet 2 Tx 64-byte Frames	E2TX64	0x2380
Ethernet 2 Tx 65- to 127-byte Frames	E2TX127	0x2384
Ethernet 2 Tx 128- to 255-byte Frames	E2TX255	0x2388
Ethernet 2 Rx 64-byte Frames	E2RX64	0x238C
Ethernet 2 Rx 65- to 127-byte Frames	E2RX127	0x2390
Ethernet 2 Rx 128- to 255-byte Frames	E2RX255	0x2394
Ethernet 2 Octet Transmitted OK	E2TXOK	0x2398
Ethernet 2 Tx Pause Frames	E2TXCF	0x239C

Table 18-10. MSC8156E QUICC Engine Register Summary (Continued)

Register Name	Acronym	Offset
Ethernet 2 Multicast Frame Transmitted OK	E2TMCA	0x23A0
Ethernet 2 Broadcast Frames Transmitted OK	E2TBCA	0x23A4
Ethernet 2 Number of Frames Received OK	E2RXFOK	0x23A8
Ethernet 2 Rx Octets OK	E2RBYT	0x23AC
Ethernet 2 Rx Octets	E2RXBOK	0x23B0
Ethernet 2 Multicast Frame Received OK	E2RMCA	0x23B4
Ethernet 2 Broadcast Frames Received OK	E2RBCA	0x23B8
Ethernet 2 Statistic Counters Carry Register	E2SCAR	0x23BC
Ethernet 2 Statistic Counters Carry Mask Register	E2SCAM	0x23C0
MIIGSK1 Enable Register	MIIGSK1_ENR	0x2808
MIIGSK2 Enable Register	MIIGSK2_ENR	0x2A08
SDMA Registers		
Serial DMA Status Register	SDSR	0x4000
Serial DMA Mode Register	SDMR	0x4004
Serial DMA Threshold Register	SDTR	0x4008
Serial DMA Hysteresis Register	SDHY	0x4010
Serial DMA Transfer Address Register	SDTA	0x4018
Serial DMA Transfer Channel Number Register	SDTM	0x4020
Serial DMA Address Qualify Register	SDAQR	0x4038
Serial DMA Address Qualify Mask Register	SDAQMR.	0x403C
Serial DMA Temporary Buffer Base in Multi-User RAM Value	SDEBCR	0x4044

TDM Interface 19

The MSC8156E Time-Division Multiplexing (TDM) interface enables communication among many devices over a single bus. Traffic is managed according to a time-division multiplexing method in which only one device drives the bus (transmit) for each channel. Each device drives its active transmit channels and samples its active receive channels when its channel is active. It is the system designer's responsibility to guarantee that there is no conflict in transmit channel allocation.

The TDM interface is composed of four identical and independent TDM modules, each supporting 256 bidirectional channels (256 transmit and 256 receive channels) running at up to 62.5 Mbps with 2-, 4-, 8-, and 16-bit word size. The TDM bus connects gluelessly to most T1 /E1 framers as well as to common buses such as the ST-Bus. Each TDM module operates in independent or shared mode when receiving or transmitting data:

- In independent mode, there are different sync, clock, and data links for receive and transmit.
- In shared sync and clock mode, the clock and the sync are shared between the transmit and receive with different data links for the receive and transmit.
- In shared data link mode, the receive and transmit share sync, clock, and full duplex data links between the transmit and receive. The clock and the sync signals can also be shared between the TDM modules.

At any time, each channel is individually set to active or inactive. An on-the-fly hardware A-law/µ–law conversion is supported for 8-bit channels. A channel is transparent or A-law/µ–law. Its data is collected in its own buffer location independently from other channel buffers. Memory space size is 16 MB for transparent channels and 32 MB for A-law/µ–law channels.

The direction of the bits in the channel (MSB first/LSB first) is configured globally for each TDM module. The direction of TSN is set to input or output. The polarity of the clock (sample/drive at clock rise or fall) is independently configured for the receiver and transmitter. The polarity of TSN/RSN/FSYN is configured to positive or negative.

The four TDM modules have an I/O matrix that routes the clock and sync signals between the TDM modules and the MSC8156E signal lines. The TDM may be configured by all six SC3850 cores (see **Figure 19-1**), as well as by an external host. Data is received and transmitted from the TDM modules to the channel buffers through the internal MBus. **Figure 19-2** shows the TDM block diagram and the receive and transmit data flows. The dashed line depicts the transmit data flow from the system I/F to the I/O matrix; the solid line depicts the receive data flow from the I/O matrix to the receive buffers on the system I/F.

Serial data received from the I/O matrix is packed and stored in the TDM local memory buffer. From the local memory buffer, the data is converted according to A/µ transformation (if needed) and re-packed for transaction to the system I/F. Data transmission occurs in a similar way but in reverse order. The channel data is transferred from the transmit data buffers being converted by the A/µ logic and stored in the TDM local memory buffer. Then the data is transmitted to the transmit serial block and to the I/O matrix.

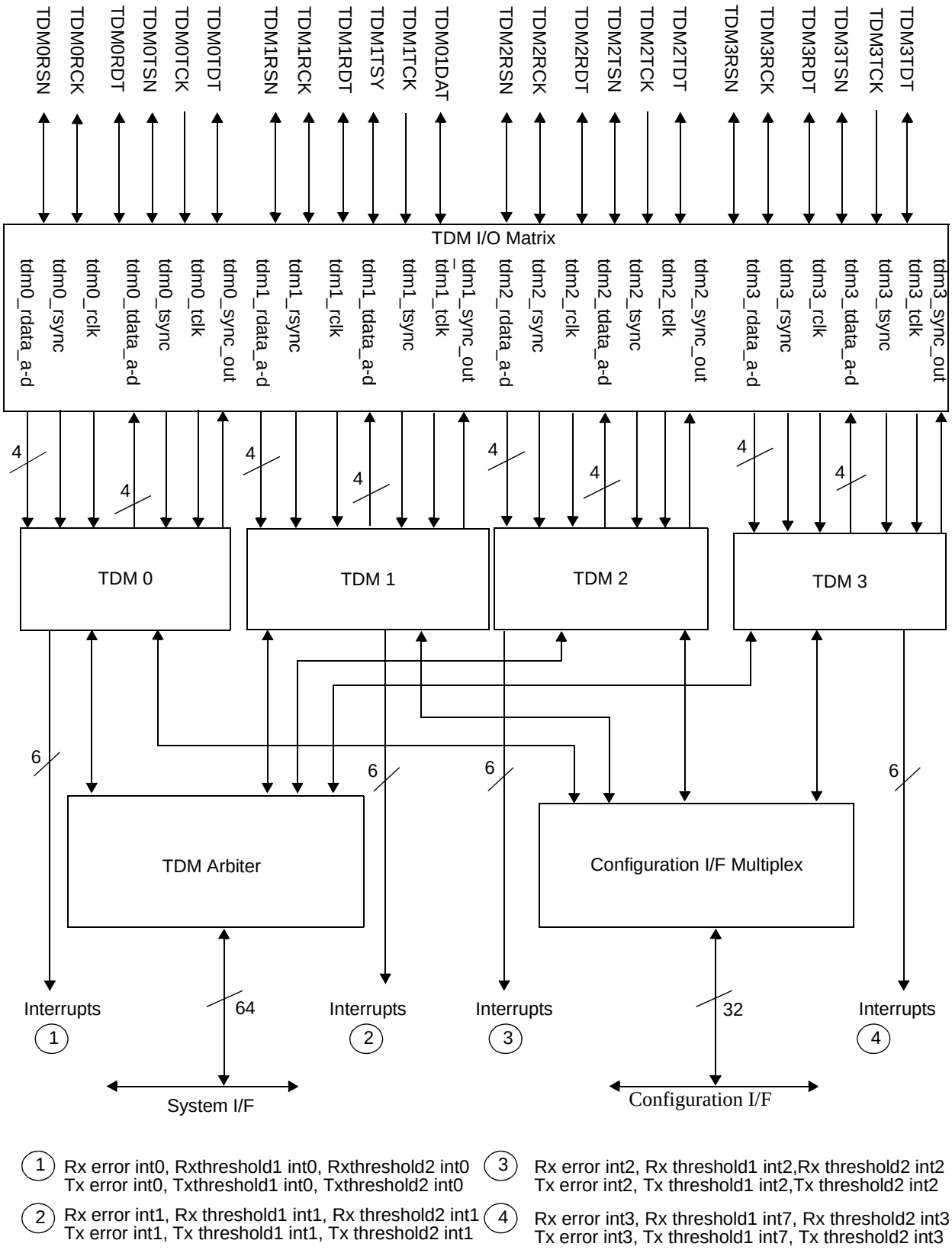


Figure 19-1. General TDM Module Interface

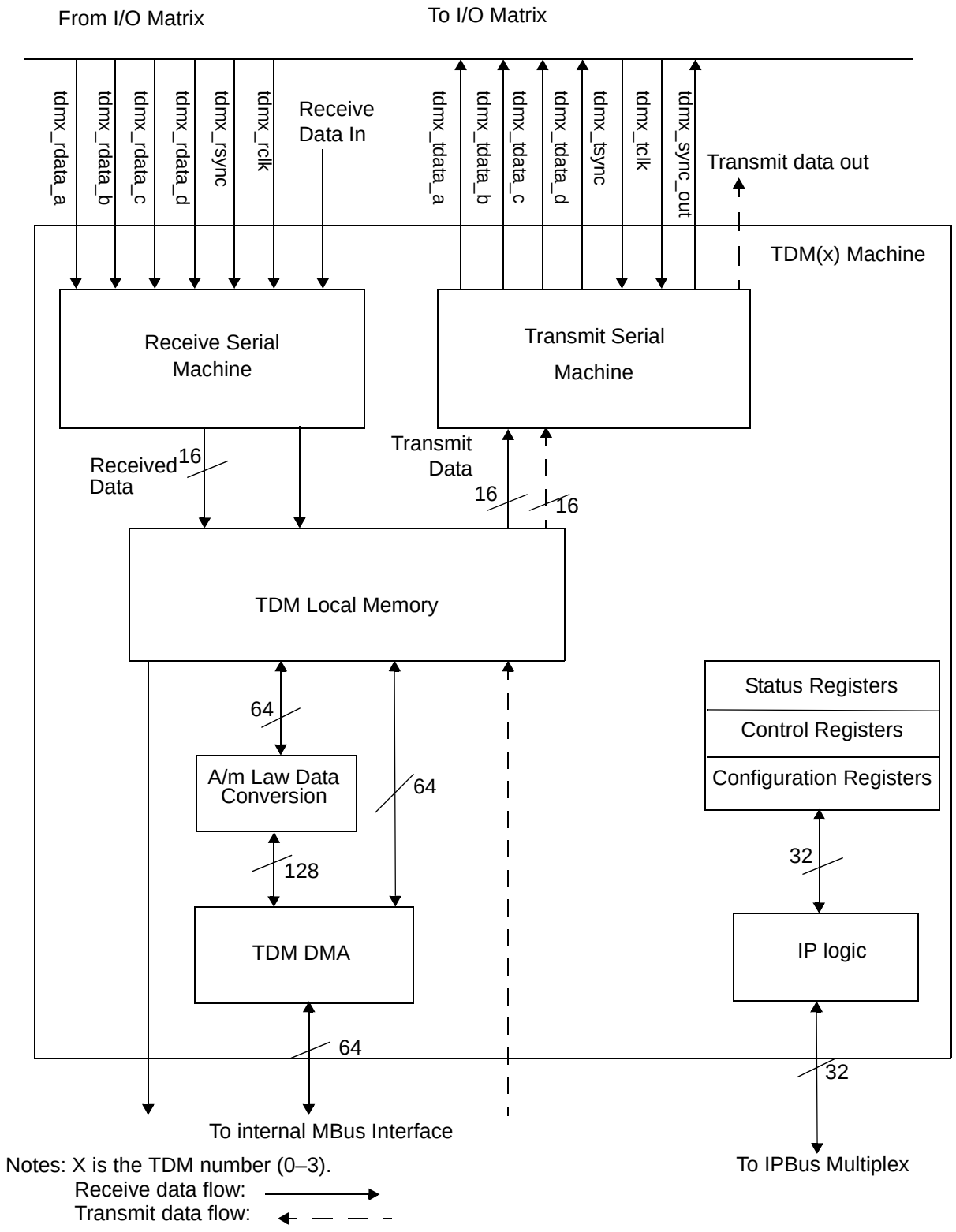


Figure 19-2. TDM Block Diagram

19.1 Typical Configurations

The TDM connects in various configurations. **Figure 19-3** shows two MSC8156E devices that connect point-to-point. Data transmits from the device on the left to the device on the right or *vice versa*.

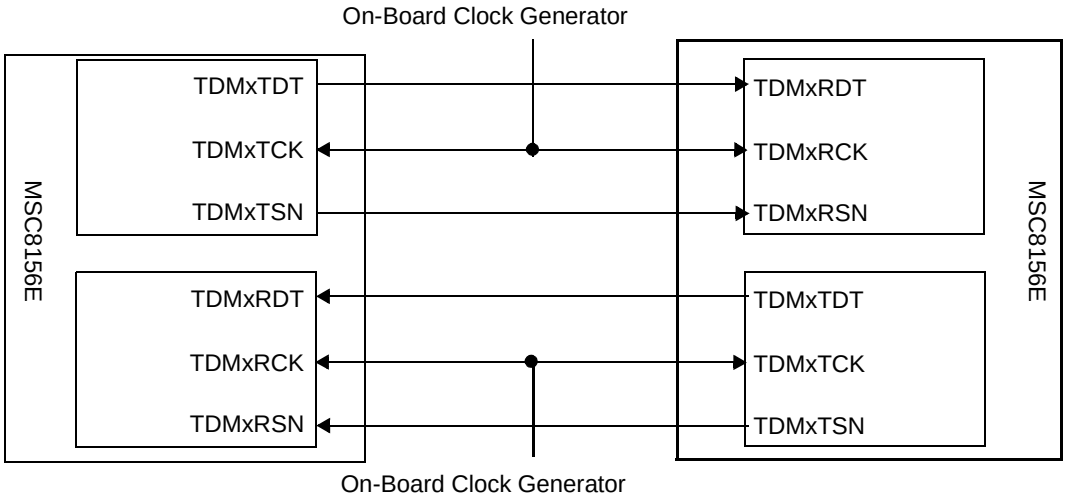


Figure 19-3. TDM Point-to-Point Configuration

Figure 19-4 depicts a TDM point to multi-point configuration. Multiple MSC8156E devices connect on the same TDM bus, which connects to the network through a framer.

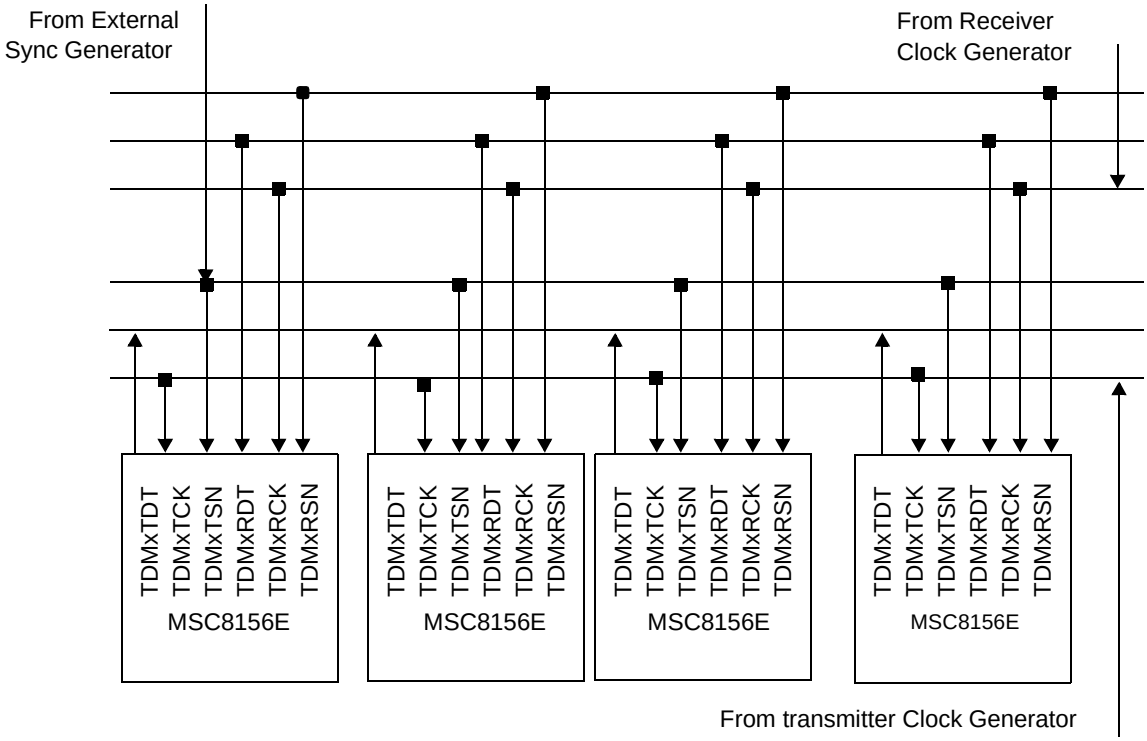


Figure 19-4. TDM Point-to-Multi-Point Configuration

Figure 19-5 depicts an application in which all the TDM modules share the sync and the clock (see **Figure 19-11**). Therefore, each TDM module supports one or two active links. In this example, 16 receive link and 16 transmit links connect to two MSC8156E devices.

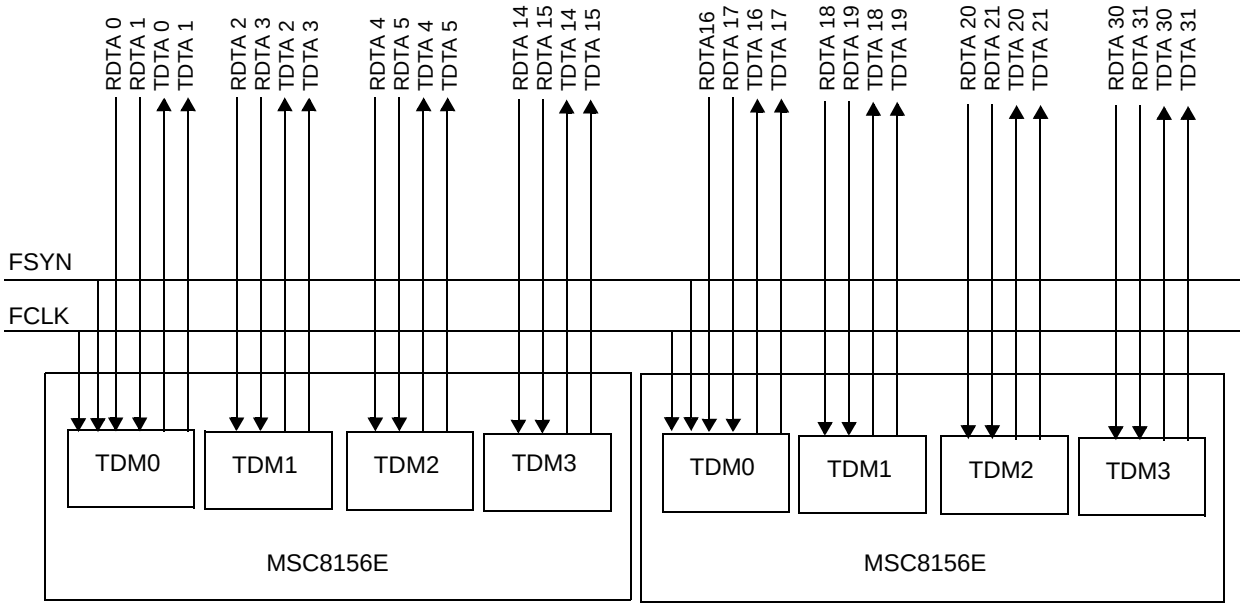


Figure 19-5. Common Frame Sync and Clock

19.2 TDM Basics

Multiple TDM channels are transferred sequentially in a structure called a frame. The frame start is identified by a frame sync signal that is briefly asserted at the beginning of every frame. Each of the four TDM modules can receive or transmit up to 256 channels at a granularity of two. The number of receive channels is determined by the RNCF field in the TDMx Receive Frame Parameters Register (TDMxRFP) (see page 19-47). The number of transmit channels is determined by the TNCF field in the TDMx Transmit Frame Parameters Register (TDMxTFP) (see page 19-50).

The size of all the channels (for each TDM module) is unified and it can be 2-, 4-, 8-, and 16 bits. The receive channel size is determined by the RCS field in the TDMxRFP; the transmit channel size is determined by the TCS field in the TDMxTFP (refer to page 19-50).

When the TDM connects to a T1 framer, the RT1 field in the TDMx Receive Frame Parameters Register (TDMxRFP) (see page 19-47) and the TT1 field in the TDMx Transmit Frame Parameters Register (TDMxTFP) (see page 19-50) should be set. The T1 frame contains 193 bits (24 channels of 8 bits each) when the first bit of the frame is a Frame Alignment bit that is not used by the TDM. At the T1 received frame, the Frame Alignment bit is removed and does not transfer to the main memory. At the transmit T1 frame, the bit is not driven out.

Figure 19-6 shows an example of TDMx frames. The receive frame contains two 2-bit channels. The transmit frame contains four 4-bit channels. **Figure 19-7** shows an example of T1 frame. In T1 mode, the first bit of the frame is not used by the TDM.

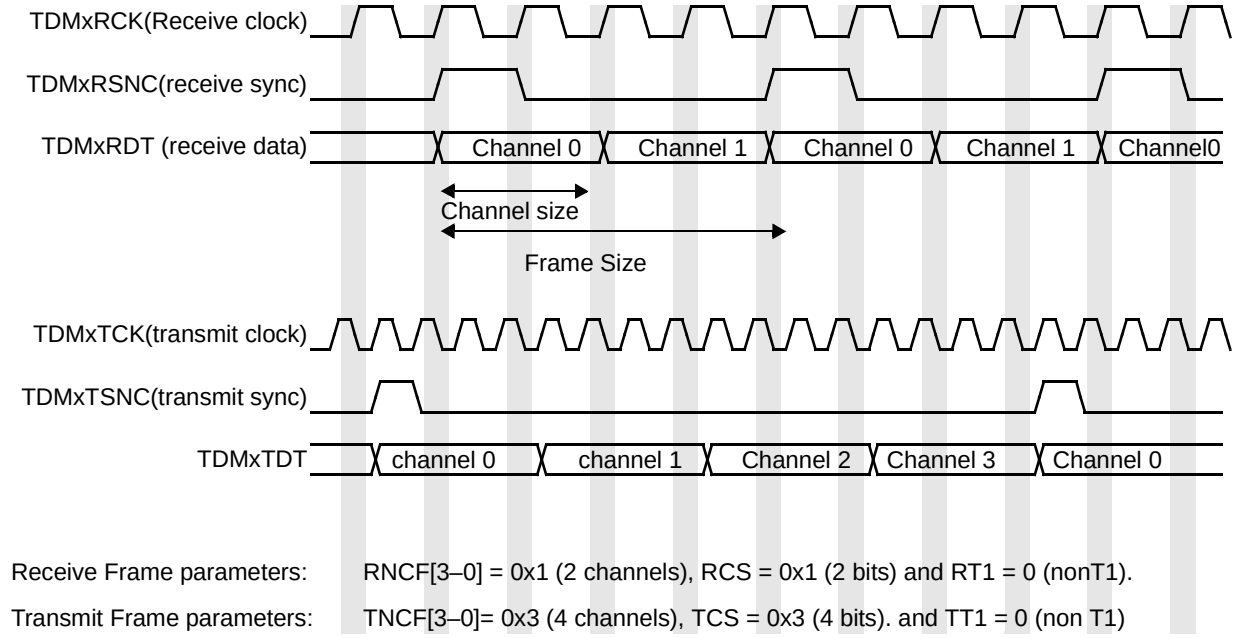


Figure 19-6. TDM Frames

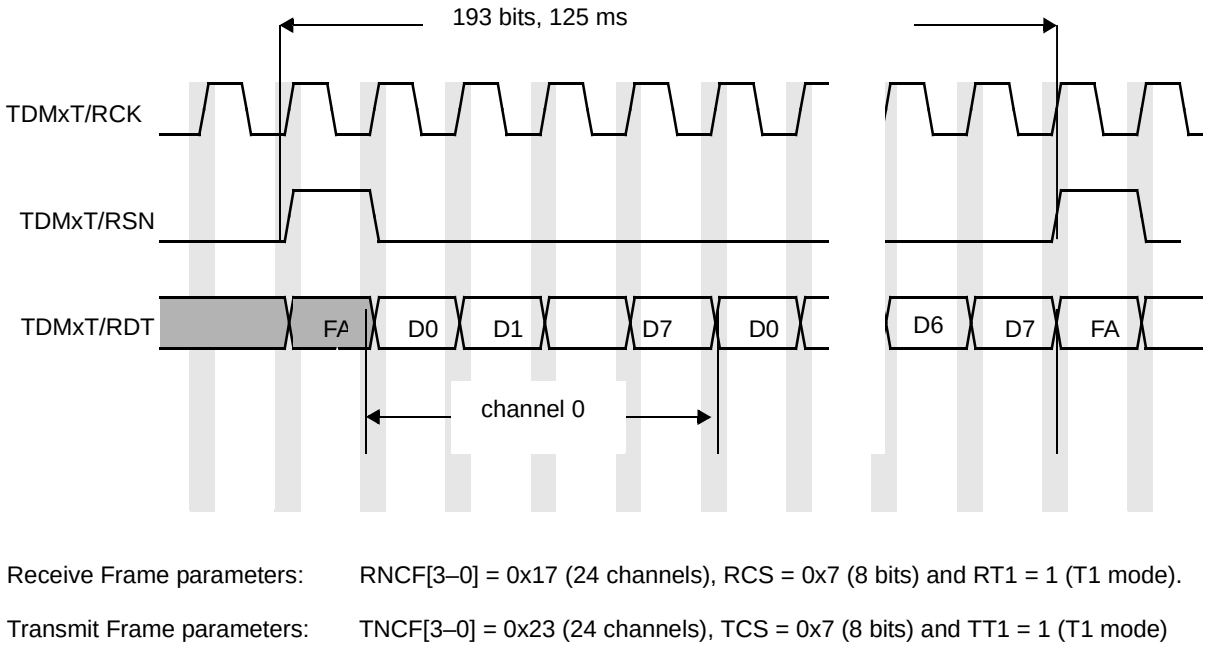


Figure 19-7. T1 Frame

19.2.1 Common Signals for the TDM Modules

The sync and clock signals can be shared among the TDM modules or separate for each TDM module. When the CTS bit of the TDMx General Interface Register (see page 19-36) is equal to 1, the TDM modules share sync and clock signals. In this mode, the common signals connect to the following signal lines:

- In non-independent mode, connect the shared sync to TDM0TSN (receive and transmit of all TDM modules share the same sync signal).
- In non-independent mode, connect the shared clock to TDM0TCK (receive and transmit of all TDM modules share the same clock signal).
- In independent mode, connect the transmit shared sync to TDM0TSNC (transmit of all TDM modules share the same sync signal). Connect the receive shared sync to TDM1TSNC (receive of all TDM modules share the same sync signal)
- In independent mode, connect the transmit shared clock to TDM0TCK (transmit of all TDM modules share the same clock signal). Connect the receive shared clock to TDM1TCK (receive of all TDM modules share the same clock signal).
- When the TDMxTIR[TSO] bit is set to a value of 1 (see page 19-45), the sync out signal drives out through TDM0TSN.

The configuration registers (see page 19-36) should be identical for the TDM modules that share signals. There are only seven possibilities for sharing TDMs:

- TDM0 and TDM1.
- TDM0, TDM1, and TDM2.
- TDM0, TDM1, TDM2 and TDM3.

Figure 19-8 illustrates a common receive sync, receive clock, transmit sync, and transmit clock for TDM0 and TDM1. When the CTS bit of the TDMx General Interface Register (see page 19-36) is cleared, the TDM modules do not share signals. In **Figure 19-8**, TDM2–TDM3 do not share signals with the other TDM modules

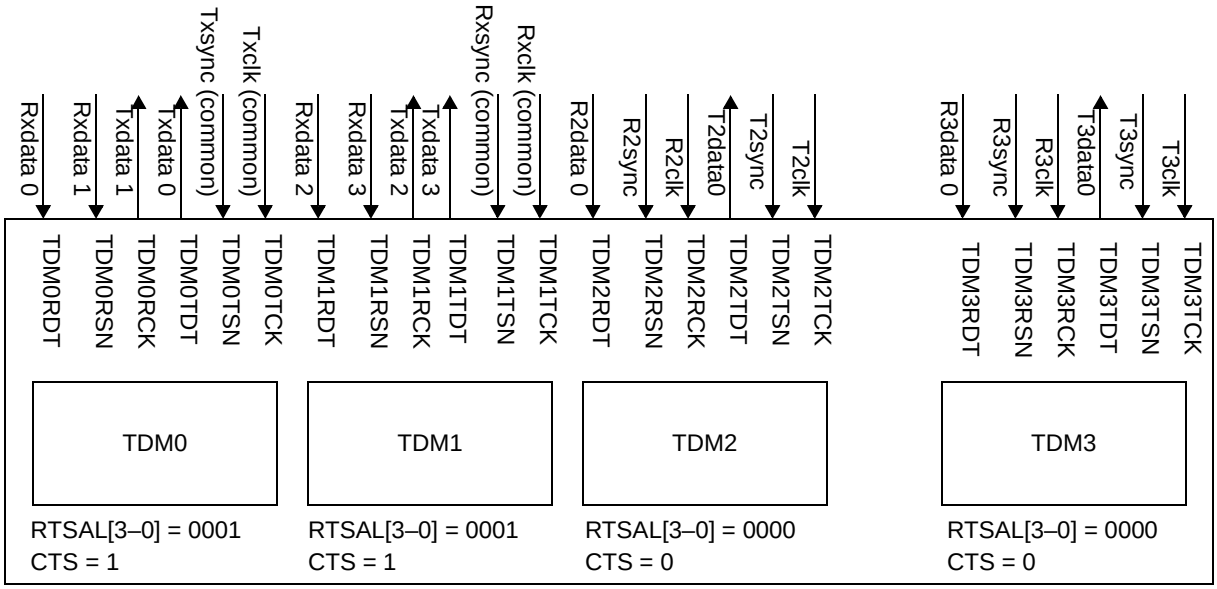


Figure 19-8. TDM Modules Model

In **Figure 19-9**, all four TDM modules share the same receive clock and sync and the same transmit clock and sync. The receive clock connects to the TDM1TCK port. The transmit clock connects to TDM0TCK, the receive sync connects to TDM1TSN, and the transmit sync connects to TDM0TSNC. Each module has two active data links. Notice that TDMxTSN, TDMxTCK when $2 \le x \le 3$ are not used.

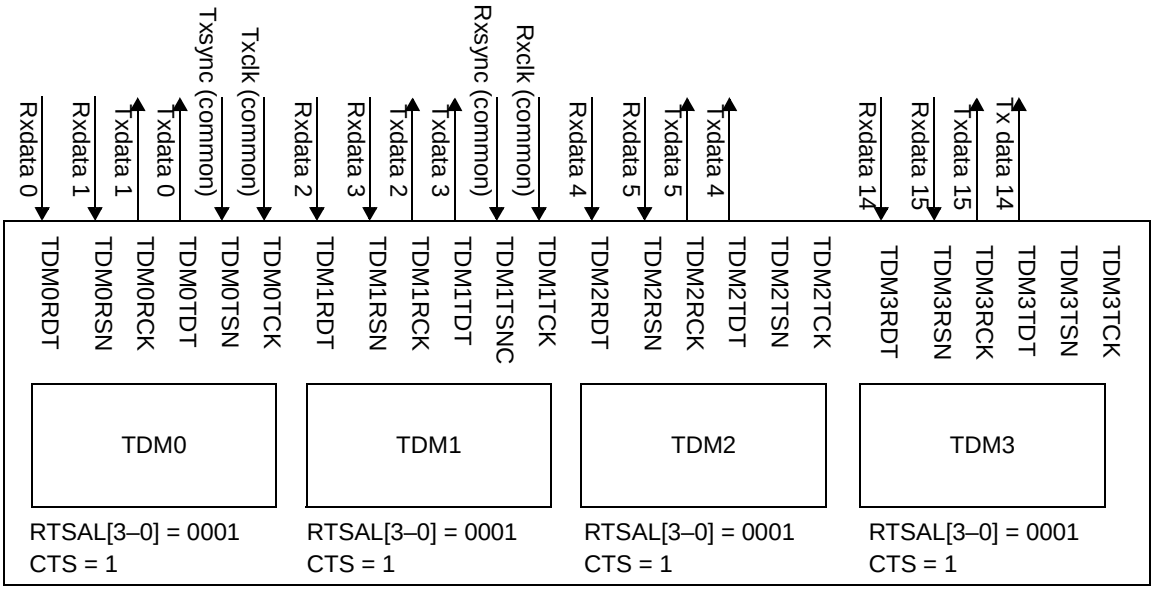


Figure 19-9. Shared Receive Sync and Clock and Transmit Sync and Clock

In **Figure 19-10**, all four TDM modules share the same frame sync, clock, and data links. Notice that TDMxTCK and TDMxTSN when $1 \leq x \leq 3$ are not used.

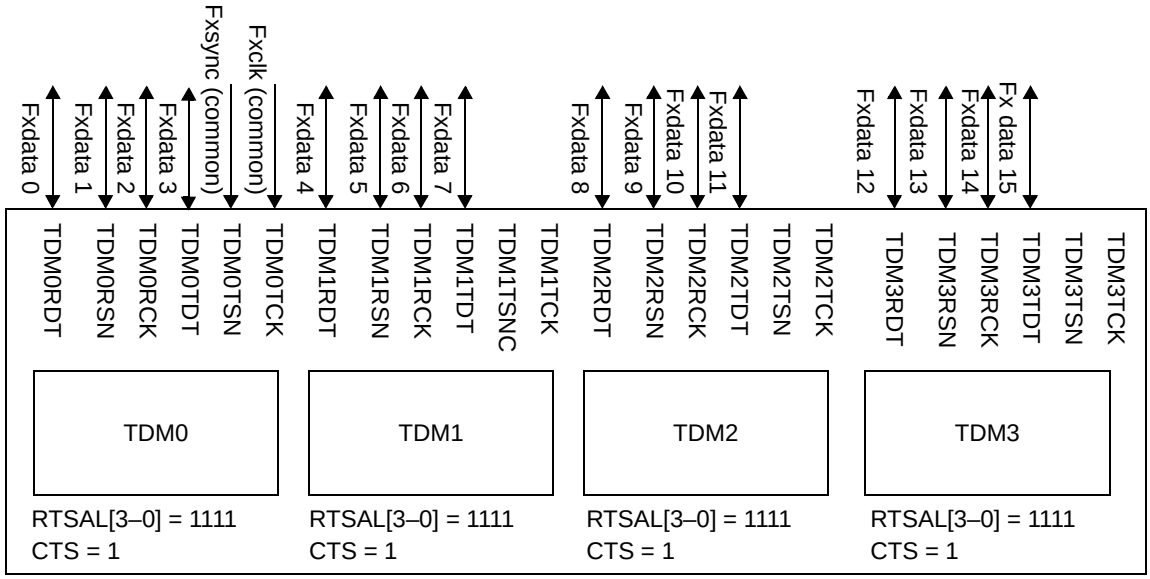


Figure 19-10. Shared Frame Sync, Clock, and Data Links

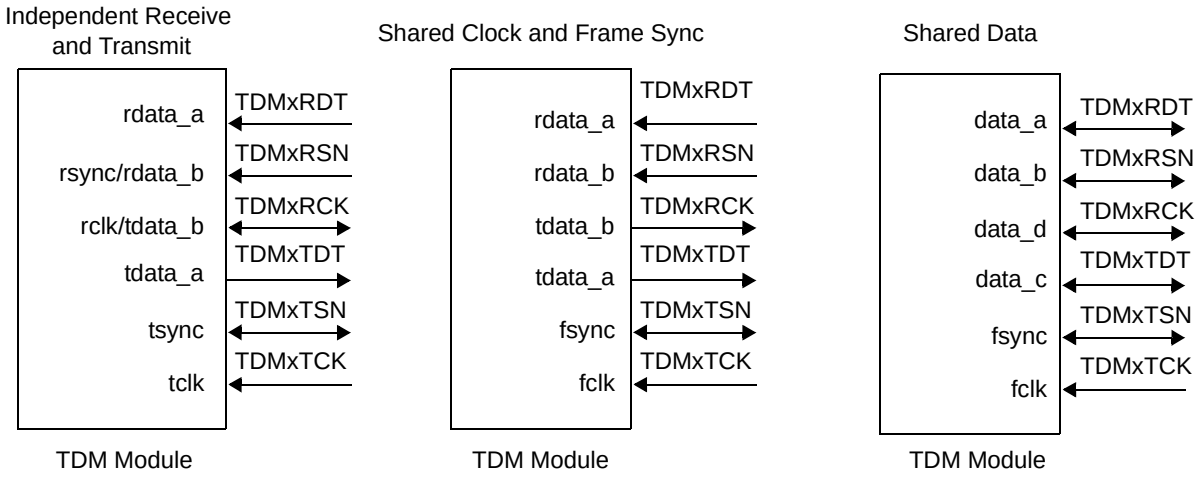
19.2.2 Receiver and Transmitter Independent or Shared Operation

The TDM operates with the transmit and receive operations running either independently or shared, as illustrated in **Figure 19-11**. When the two most significant bits of the RTSAL field (RTSAL[3–2]) in the TDMx General Interface Register (see page 19-36) equal 0b00, the receive and the transmit are independent as illustrated on the left side of **Figure 19-11**. In this mode, there is one input receive data link and one output transmit data link. If the TDM shares signals with other TDM modules (CTS = 1), it can receive two data links and it can output two data links.

When the RTSAL[3–2] in the TDMx General Interface Register (see page 19-36) equal 0b01, the receive and transmit are shared as illustrated in the middle of **Figure 19-11**. The transmit and the receive share the Frame Sync (FSYN) and the Frame Clock (FCLK) signals. The number of receive and the transmit active links can be one or two. The direction of the receive links is input, and the direction of the transmit links is output.

When RTSAL[3–2] in the TDMx General Interface Register (see page 19-36) equal 0b11, the receive and the transmit are shared as illustrated on the right side of **Figure 19-11**. The transmit and the receive share the Frame Sync (FSYN), the Frame Clock (FCLK), and the data signals. In this mode, the data links are full duplex and are used for both transmit and receive, so the number of active links can be 1, 2, or 4.

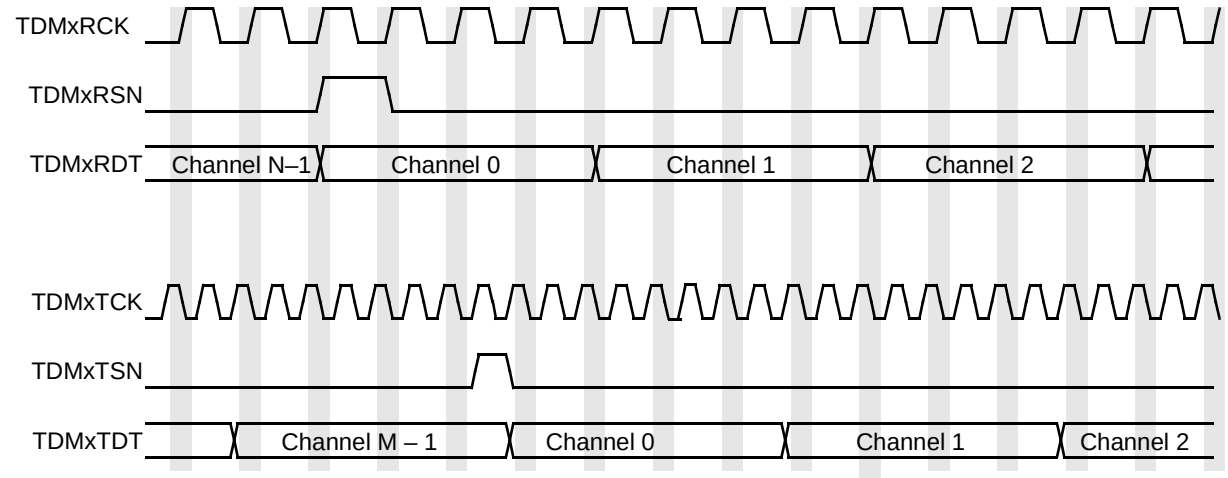
When RTSAL [1–0] equals 0b11, there are four active links: DATA_A, DATA_B, DATA_C, and DATA_D. When RTSAL[1–0] equals 0b01, there are two active links: DATA_A and DATA_B. When RTSAL[1–0] equals 0b00, there is one active link, DATA_A.



x Defines the TDM number.
FSYNC (frame sync) specifies that the receiver and transmitter share the same sync.
FCLK (frame clock) specifies that the receiver and transmitter share the same clock.

Figure 19-11. TDM Module Sharing Modes

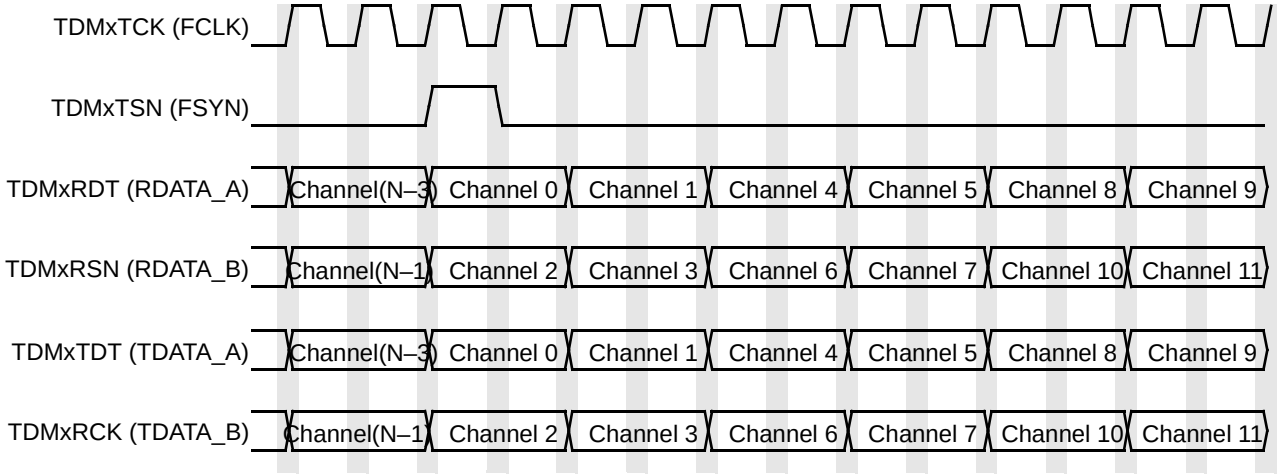
Figure 19-12 describes the TDM interface when the receive and transmit are totally independent (TDMxGIR[RTSAL] = 0b0000). The TDMxRCK is not synchronized to the TDMxTCK. They differ according to the sync location relative to the beginning of the frame and the number of bits.



X The TDM number.
N The number of channel in the receive TDM frame.
M The number of channels in the transmit TDM frame.

Figure 19-12. Receive and Transmit Totally Independent

Figure 19-13 describes the TDM timing interface when TDMxGIR[RTSAL] = 0b0101. The frame sync and the clock is shared between the receive and transmit, and links A and links B are active. RDT and RSN are used as received data links, and TDT and RCK are used as transmit data links. Channels are organized in pairs: channel i and channel i+1 are always received/transmitted on the same link, one after another.

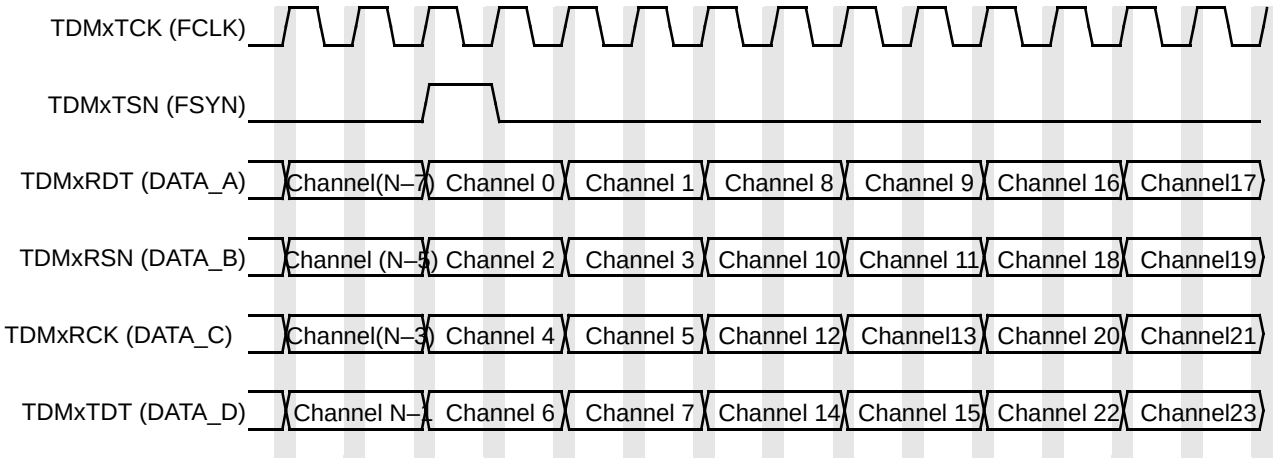


x The TDM number.

N The number of channels in a TDM frame.

Figure 19-13. Shared Sync and Clock (Two Active Data Links)

Figure 19-14 shows the TDM timing interface when the RTSAL field is set to 0b1111.The frame sync, the clock, and the data links are common. All four data links are active and are used for both transmit and receive.



x The TDM number.

N Number of channels in a TDM frame.

The data links are bidirectional.

The clock and the sync are common.

Figure 19-14. Receive and Transmit Share Sync, Clock, and Data (Four Active Links)

Note: The number of channels in a frame must be a multiple of 2 × the number of data links.

Table 19-1 shows the number of channels each active link supports.

Table 19-1. Maximum Number of Channels Per Active Link

Number of Active Links	1 Active Link	2 Active Links	4 Active Links
Maximum channel number per active link in one TDM module	256	128	64

The TDM bit rate depends on:

- *System bus clock*. The TDM processes the data using the CLASS clock rate divided by 2, so the maximum data bit rate is limited to one half of the rate of the CLASS clock.
- *Number of active links*. The total bit rate is shared by all active links, so one active link supports the highest bit rate.
- *Channel width*. When there are more bits per channel, there are fewer channels per second, so higher bit rates can be processed per second.

Table 19-2 describes the maximum bit rate as a function of these parameters. Factors other than the width of the channel can affect the bit rate, such as the memory system load, for example.

Table 19-2. Factors Affecting Maximum Bit Rate

Channel Width (Bits)	1 Active Link	2 Active Links	4 Active Links
2	(half the CLASS frequency)/8	(half the CLASS frequency)/12	(half the CLASS frequency)/20
4	(half the CLASS frequency)/4	(half the CLASS frequency)/6	(half the CLASS frequency)/10
8	(half the CLASS frequency)/2	(half the CLASS frequency)/3	(half the CLASS frequency)/5
16	(half the CLASS frequency)/2	(half the CLASS frequency)/2	(half the CLASS frequency)/2.5

Note: In addition to the limits defined in **Table 19-2**, the maximum serial frequency is limited to 62.5 MHz.

19.2.3 TDM Data Structures

TDM data structures are stored in transmit and receive local memory, as follows:

- *TDM receive local memory*. Received data is stored in 256 8-byte entries located in addresses between 0x0000–0x07FF, which is offset from the TDMx receive local memory (see **Chapter 9**, *Memory Map*). This memory contains 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the received data is indicated in the RNB field of the TDMx Receive Number of Buffers Register (TDMxRNB) (discussed on page 19-67). Channel C in buffer B is the 8 bytes starting at (256 / (RNB + 1) × B + C) × 8.

- *TDM transmit local memory*. Transmit data is located in the TDM local memory before it is transmitted externally. The data is stored in 256 8-byte entries in addresses between 0x1800– 0x1FFF, which is offset from the TDMx receive local memory (see **Chapter 9**, *Memory Map*). This memory can contain 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the transmitted data is indicated in the TNB field of the TDMx Transmitter Number of Buffers Register (TDMxTNB). Channel C in buffer B is the 8 bytes starting at (256 / (TNB + 1) × B + C) × 8.

Figure 19-15 shows an example of TDM local memory that contains four transmit buffers and one receive buffer. Up to 32 transmit bytes of channel 2 are located in four buffers (TNB = 3). Only 8 receive bytes of channel 2 are located in one buffer (RNB = 0). Each buffer contains 8 bytes per channel.

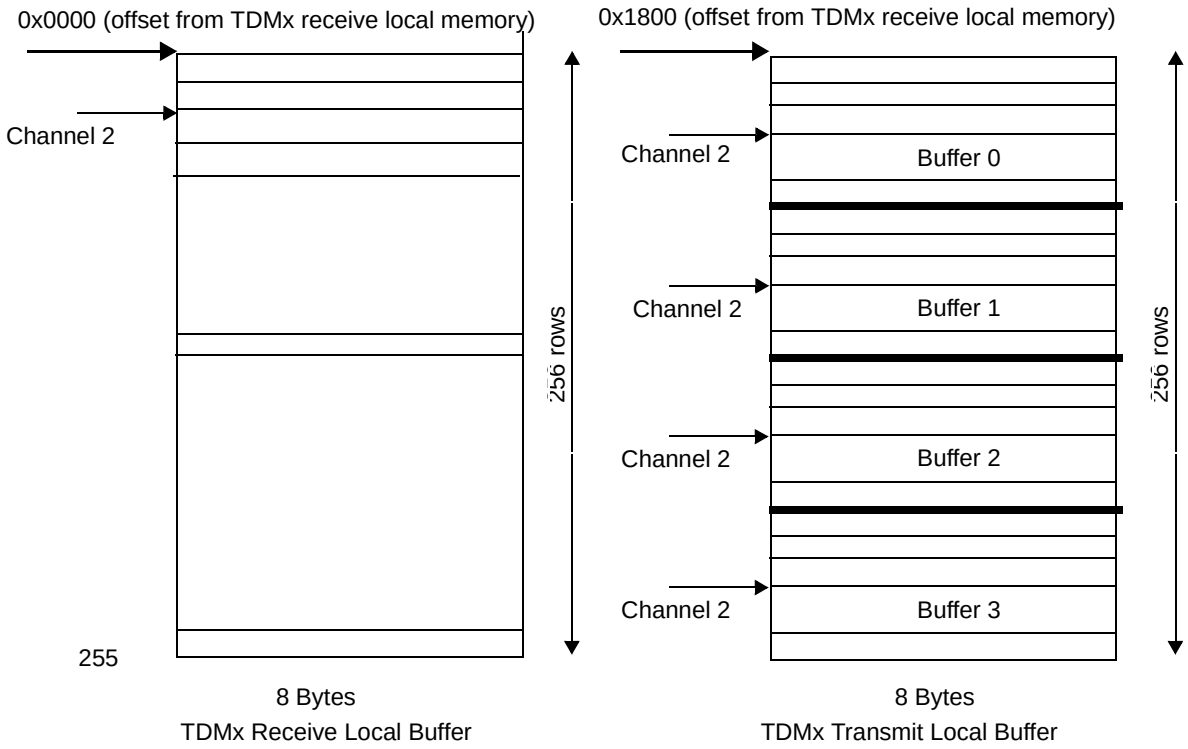


Figure 19-15. TDM Local Buffer (Receive and Transmit)

When the TDM transmit local memory is accessed using addresses with 8-byte alignment, data is written to the 4 LSB of the memory row. **Figure 19-16** describes the TDMx local memory after write access of 0x01234567 to address 0x1800 (offset from TDMx Receive Local Memory) and 0x89ABCDEF to address 0x1804 (offset from TDMx Receive Local Memory). If TDMxTIR[TBOR] = 1, the 0x89ABCDEF data is transmitted before the 0x01234567 data.

Figure 19-16. TDMx Local Memory Write Example

When the TDM receive local memory is accessed using addresses with 8-byte alignment, data is read from the 4 LSB of the memory row. **Figure 19-17** describes a row in the TDMx local memory, in which the 0x00112233 data is received before the 0x44556677 data (if TDMxRIR[RBOR] = 1). In this example, the data to be read from address 0x1000 (offset from TDMx Receive Local Memory) is 0x44556677, and the data to be read from address 0x1004 (offset from TDMx Receive Local Memory) is 0x00112233.

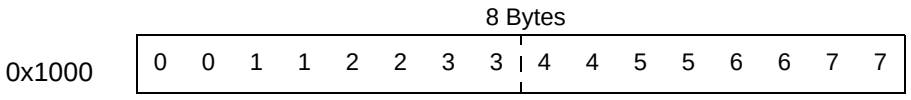


Figure 19-17. TDMx Local Memory Read Example

19.2.4 Serial Interface

This section covers issues related to the serial interface, such as how to configure the frame sync and how to control the data order of the bits in the channel.

19.2.4.1 Sync Out Configuration

TDMxTSN is programmed as either an input or output by writing 1 to the TSO bit in the Transmit Interface Register (TDMxTIR) (see page 19-45). When the TSO bit value is equal to 1, the sync_out signal connects to the sync out signal in the TDM I/O matrix and is output via TDMxTSN. When the TDM modules share a sync and clock signals (the CTS bit is set), then the TDMx[TSO] bits should be equal for all the TDM modules and they determine whether the sync arrives from the board or is generated by the TDM0 transmitter. Configuring the sync out signal involves the parameters listed in **Table 19-3**.

Table 19-3. Parameters in Configuring the Frame Sync (TDMxTIR[TSO] = 1)

Task	Register
Control the length of the sync_out signal. If the SOL bit is clear then the sync_out width is one transmit bit, else the sync_out length is one transmit channel.	TDMxTIR[SOL], page 19-45
Control the transmit clock edge on which the sync_out is driven out. If the SOE bit is clear, the sync_out is driven out on the rising edge of the transmit clock.	TDMxTIR[SOE], page 19-45
Control the sync_out level. The sync out level must be identical to the transmit sync. It is determined by the TSL configuration field.	TDMxTIR[TSL], page 19-45
Control the sync_out distance. The distance between two consecutive sync out events is constant and equal to one transmit frame. The transmit frame length is determined by the transmitter configuration fields TCS,TNCF, TT1 and RTSAL[1–0]. The distance is = (TCS + 1) × (TNCF + 1) / (RTSAL[1–0] + 1) + TT1.	TDMxTFP[TNCF], page 19-50 TDMxTFP[TCS], page 19-50 TDMxTFP[TT1], page 19-50 TDMxGIR[RTSAL], page 19-36

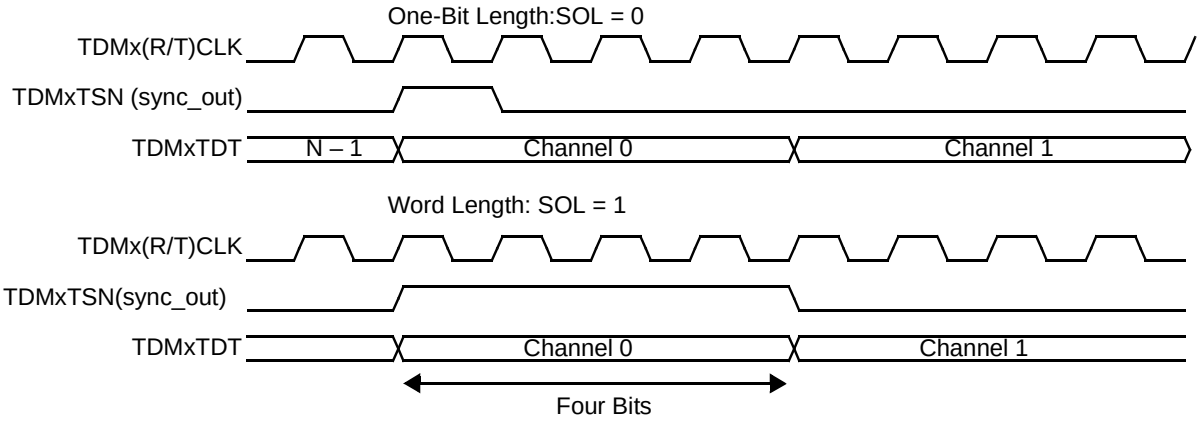


Figure 19-18. Sync Length Selection

19.2.4.2 Sync In Configuration

TDMxRSN is an input that identifies the beginning of the received frame. TDMxTSN can be an input or output from the TDM, but the transmitter refers to the transmit sync as an input because the connection between the sync_out signal and the transmit sync (tsync) occurs only in the TDM I/O matrix. **Figure 19-19** illustrates the relation between the data, the sync, and the clock for various configurations. The receive data and frame sync are sampled with the rising or falling edge of the receive clock. The transmit frame sync is sampled with the rising or falling edge of the transmit clock. The transmit data drives out at the rising or falling edge of the transmit clock. The delay between the first data bit of the frame and the sync is referred to as the rising edge of the sync. **Table 19-4** lists the frame sync controls.

Table 19-4. Transmit and Receive Frame Configuration

Control	Register
Which receive clock edge samples the receive frame sync. If RFSE is clear, the receive frame sync is sampled on the rising edge of the receive clock.	TDMxRIR[RFSE] bit, page 19-43.
Which transmit clock edge samples the transmit frame sync. If TFSE is clear, the transmit frame sync is sampled on the rising edge of the transmit clock.	TDMxTIR[TFSE] bit, page 19-45
Which receive clock samples the receive data. If RDE is clear, the receive data is sampled on the rising edge of the receive clock.	TDMxRIR[RDE] bit, page 19-43
Which transmit clock edge drives out the data. If TDE is clear, then the transmit data is driven out on the rising edge of the transmit clock.	TDMxTIR[TDE] bit, page 19-45
Determines the receive sync level. If RSL is clear the receive sync level is high.	TDMxRIR[RSL] bit, page 19-43
Determines the transmit sync level. If TSL is clear the transmit sync level is high.	TDMxTIR[TSL] bit, page 19-45
Determines the timing of the receive frame sync signal relative to the first data bit of the receive frame.	TDMxRIR[RFSD] field, page 19-43
Determines the timing of the transmit frame sync signal relative to the first data bit of the transmit frame.	TDMxTIR[TFSD] field, page 19-45

The receive delay when the receive sync and the receive data are not sampled at the same clock edge is **RFSD** + 0.5. The transmit data can be driven out before the transmit sync sample. Therefore, the transmit delay when the transmit sync and transmit data are sampled/driven out at the same clock edge is (**TFSD** – 1). And when the sync and the data sampled/driven out at different clock edge is (**TFSD** – 1 + 0.5).

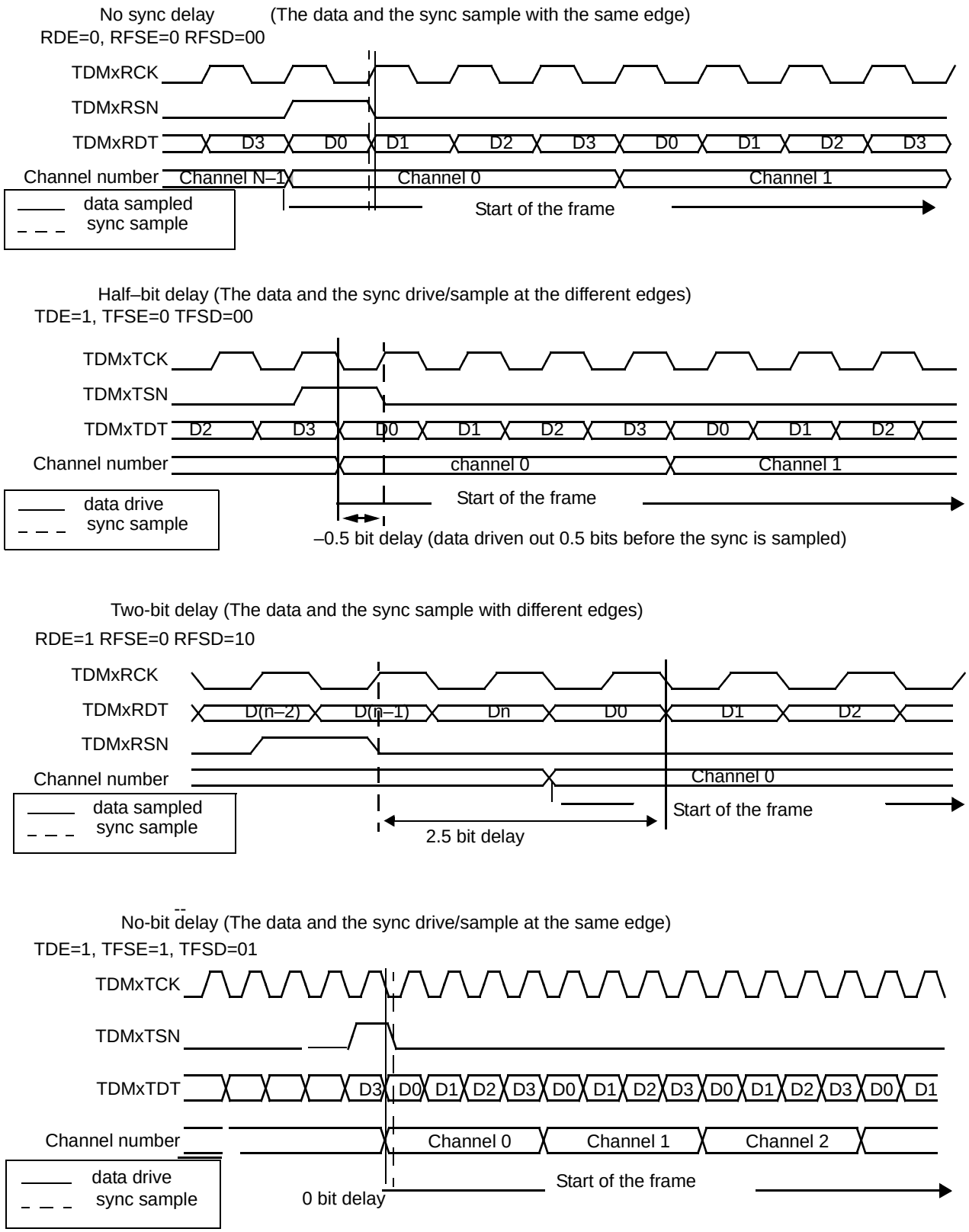


Figure 19-19. Frame Sync Configurations

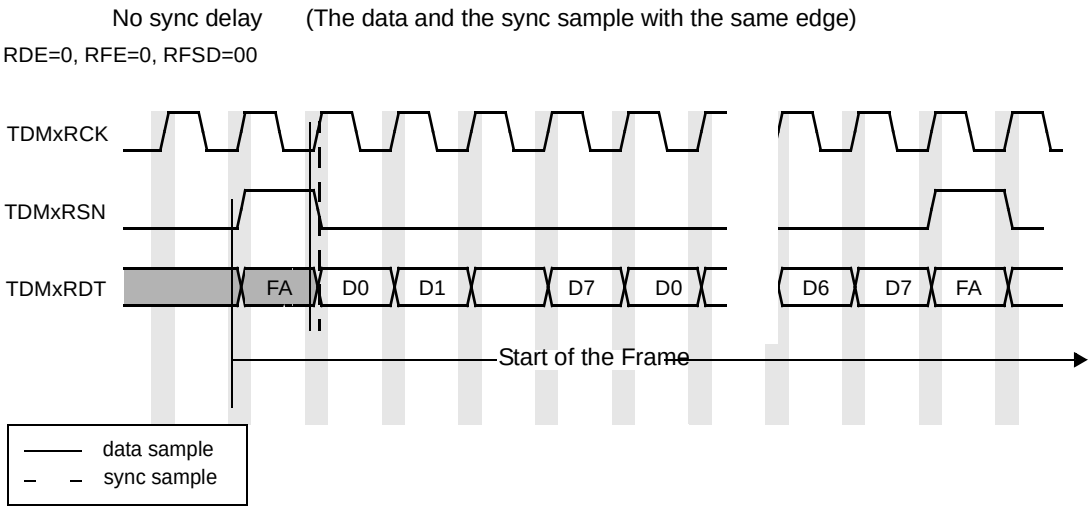


Figure 19-20. Frame Sync Configuration At T1 Mode

Figure 19-21 illustrates how the polarity of the receive sync and the transmit sync signals is controlled by the TDMxRIR[RSL] and the TDMxTIR[TSL] bits.

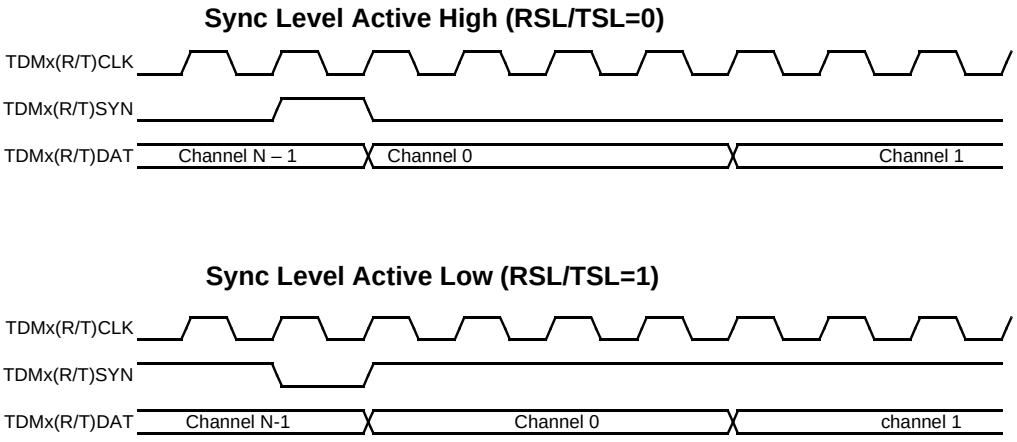


Figure 19-21. Frame Sync Polarity

19.2.4.3 Serial Interface Synchronization

The TDM module enables communication among many devices over a single bus. The receive and transmit of each TDM frame is identified by a frame sync signal that is asserted at the beginning of every frame. The frame sync synchronization is necessary when more than one device drives the bus. **Figure 19-22** shows the state diagram of the frame sync synchronization.

The details of the state diagram are as follows:

- *HUNT* (0b00). A sync event is constantly sought. As soon the sync event is detected, the state machine changes to a WAIT state. During the Hunt state, data is neither received nor transmitted.
- *WAIT* (0b01). At least one sync has been detected. The next sync event is accepted after one TDM frame. If the sync appears in the correct position, the state changes to the PRESYNC state (0b11). If the sync does not appear, the state returns to the hunt state. During the WAIT state, data is neither received nor transmitted.
- *PRESYNC* (0b11). Two sync events have been detected and the distance between the syncs is one TDM frame. If the sync event is recognized early, the state returns to the WAIT state. Otherwise, the machine transfers to the SYNC state at the last bit of the TDM frame. During PRESYNC state, data is neither received nor transmitted.
- *SYNC* (0b10). At least one sync event has appeared exactly where it was expected. This state is maintained as long as the sync event continues to appear where expected. If a sync is missed or a sync event is recognized early, the state changes to the HUNT state (0b00). During the SYNC state, data is both received and transferred.

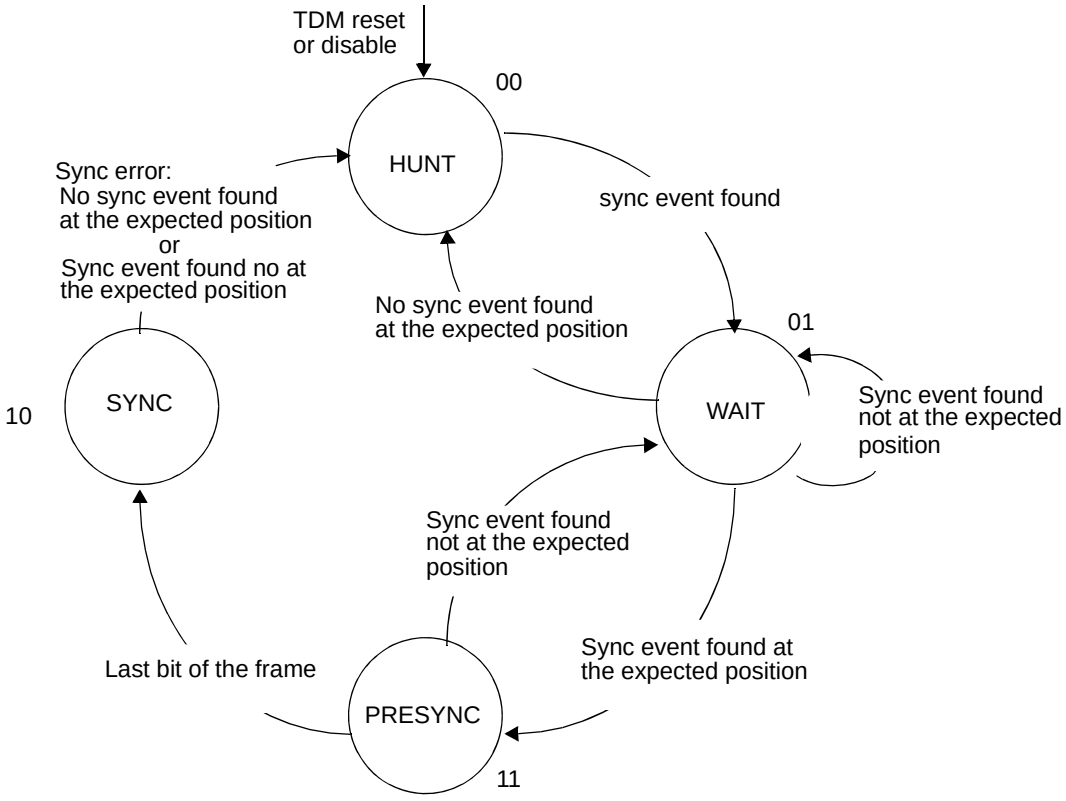


Figure 19-22. Frame Sync Synchronization State Diagram

The TDM receiver synchronizes on the receive frame sync (rsync). The state of the receive frame sync synchronization is indicated by the TDMxRSR[RSSS] field (see page 19-71). During the HUNT, WAIT, and PRESYNC states, the received data is not transferred to the buffers in the main memory for processing. When the receive sync synchronization is lost, the state transfers from SYNC to HUNT (the TDMxRER[RSE] bit is asserted) (see page 19-68). If the TDMxRIER[RSEEE] bit (see page 19-63) is also set, a receive error interrupt is generated. The interrupt service routine (ISR) should clear the TDMxRER[RSE] bit by writing a 1 to the bit before clearing the related status bit in the EPIC and before returning from the ISR.

The transmit frame sync synchronization state is indicated by the TDMxTSR[TSSS] field (see page 19-72). During the HUNT, WAIT, and PRESYNC states, new data is not driven out. If the Transmit Always Out (TDMxTIR[TAO]) field (see page 19-45) is set, then the last data is driven out until the frame sync synchronization state returns to SYNC state. If the TDMxTIR[TAO] bit is clear, data is not driven out and TDMxTDT is tri-stated. When the transmit sync synchronization is lost, the TDMxTER[TSE] bit (see page 19-69) is asserted. If the TDMxTIER[TSEIE] bit (see page 19-64) is also set, a transmit error interrupt is generated. The ISR should clear the TDMxTER[TSE] bit by writing a 1 to the bit before clearing the related status bit in the EPIC and before returning from the ISR.

The frame sync synchronization state can identify different problems. In the initial design stages, the frame sync summarization state indicates whether the TDM programming matches the actual TDM stream. During operation, the synchronization state and the error interrupts may indicate errors in the TDM module signal processing.

19.2.4.4 Reverse Data Order

Figure 19-23 illustrates how the bit order of the stored data relates to the bit order of the receive or the transmit data. The TDMxRIR[RRDO] bit defines how the receive channel data is stored in memory. If TDMxRIR[RRDO] is clear, the first bit of the received channel data is stored as the most significant bit. The TDMxTIR[TRDO] bit selects the transmit data bits order. If TDMxTIR[TRDO] is clear, the most significant bit of the memory is transmitted as the first transmit data.

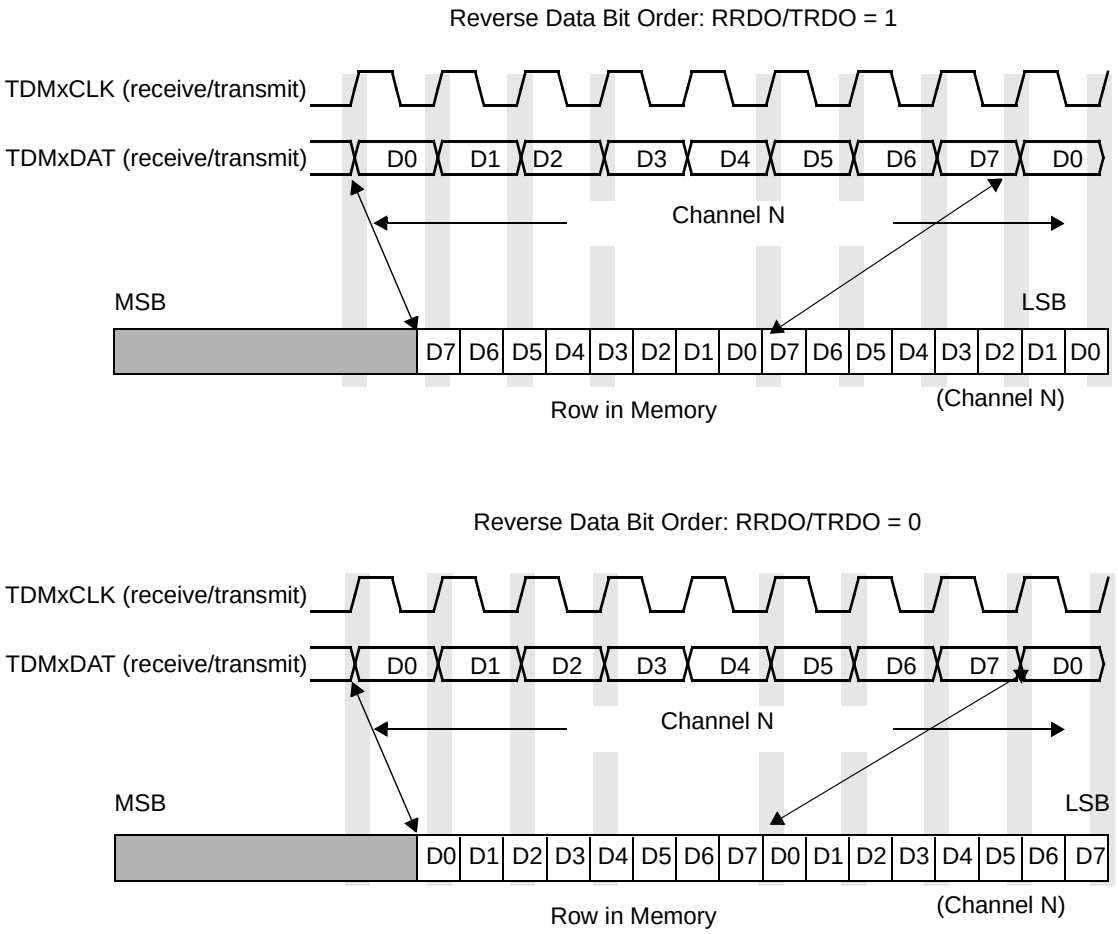


Figure 19-23. Reserve Bit Order

19.2.5 TDM Local Memory

Received data is temporarily stored in the TDM receive local memory until it is transferred to the receive buffers mapped on the system I/F. A single data transfer from TDM local memory to a memory-mapped region on the system I/F transfers at least 64 bits of data. Each channel can store more than 64 bits before the data is written to the buffers mapped on the system I/F. The TDMxRFP[RCDBL] field (see page 19-47) provides an upper boundary on the number of receive bits that can be stored in the TDM local memory. The receive data latency is defined as the time between receiving data and the time when it is available for processing by an SC3850 core. Reducing the TDMxRFP[RCDBL] value reduces the receive data latency. However, reading the TDM local memory imposes more strict latency requirements on the system I/F. The maximum receive data latency is calculated as follows: RCDBL /RCS × receive frame time.

When the amount of received data exceeds the size of the TDM receive local memory, the TDMxRER[OLBE] bit is set (see page 19-68). If the TDMxRIER[OLBEE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the system I/F and therefore cannot write the data into the destination memory (the data buffer).

Data transmitted from memories mapped on the system I/F is temporarily stored in the TDM transmit local memory until it is transferred externally. A single data transfer from the system I/F to TDM local memory transfers at least 64 bits of data. Each channel can store more than 64 bits before it is transmitted externally. The TDMxTFP[TCDBL] field provides an upper boundary on the number of transmit bits that can be stored in TDM local memory. The transmit data latency is defined as the time between when the data is read from the buffers mapped to the system I/F and when it is transmitted externally. Reducing the TDMxTFP[TCDBL] value reduces the transmit data latency. However, writing the TDM local memory imposes more strict latency requirements on the internal MBus. The maximum transmit data latency is calculated as follows: TCDBL / TCS × transmit frame time.

When the TDM cannot transfer data from data buffers to TDM local memory, an underrun occurs. When the TDM transmit local memory is empty, the TDMxTER[ULBE] bit (see page 19-69) is set and the TDMxTIER[ULBEE] bit is also set, an error interrupt is generated. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the system I/F and therefore cannot read the data from the source memory into TDM transmit local memory. The minimum latency is achieved when the RCDBL/TCDBL field is clear (only 64 bits are stored in the TDM local memory). For example, the minimum latency for a T1 application with 8 bits per channel and a frame length of 125 µs is equal to 1 ms. T1 minimum latency= 64/8 × 125 µs.

19.2.6 Buffers Mapped on System Memory

Each receive or transmit data channel is stored in a different buffer mapped on the internal system memory. This buffer can be located in any of the internal memories (such as the M2 or M3 memory) that are shared by all the SC3850 cores.

19.2.6.1 Data Buffer Size and A/μ-law Channels

Data buffer size is identical for all receive channels belonging to a TDM module and is indicated in the TDMxRDBS[RDBS] field. Data buffer size is also identical for all the transmit data buffers and is indicated in the TDMxTDBS[TDBS] field (see page 19-53). An exception is the A/μ-law channels (buffer size × 2). When the TDMxRCPRn[RCONV] field (see page 19-61) indicates that a channel is an A-law channel, the received 8 bits are converted into a 13-bit PCM sample padded with three zeros on the right. This channel therefore occupies 16 bits per 8 received bits, essentially occupying double the size. When the TDMxRCPRn[RCONV] field indicates that a channel is a μ-law channel, the received 8 bits are converted into a 14-bit PCM sample padded with two zeros on the right. This channel also occupies 16 bits per 8 received bits, essentially occupying double the size.

When the TDMxTCPRn[TCONV] field (see page 19-62) indicates that a channel is an A-law channel, the transmitted 13 bits are converted into an 8-bit PCM sample. This channel therefore occupies 16 bits (13 bits padded with three zeros at the right) per 8 transmit bits, essentially occupying double the size. When the TDMxTCPRn[TCONV] field indicates that a channel is a μ-law channel, the received 14 bits are converted into an 8-bit PCM sample. This channel also occupies 16 bits (14 bits padded with two zeros at the right) per 8 transmit bits, essentially occupying double the size. The A/μ-law conversion is performed according to the ITU-T recommendation G.711.

Note: The minimum buffer size for both transmit and receive is 16 bytes (that is, the RDBS/TDBS value is 0x00000F). The maximum buffer size for both transmit and receive is 16 MB (that is, the RDBS/TDBS value is 0xFFFFFF), but it can be further limited by the main memory size according to the number of channels in the frame.

Figure 19-24 shows how the samples are stored in the receive main data buffer (if the receive channel is A/μ law and TDMxRIR[RBOR] = 1) or the transmit main data buffer (if the transmit channel is A/μ law and TDMxTIR[TBOR] = 1).

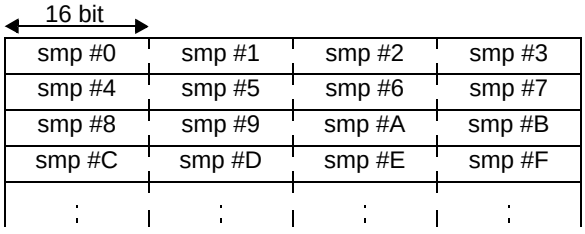


Figure 19-24. Receive/Transmit Main Data Buffer For A-Law/μ-Law Channel

19.2.6.2 Data Buffer Address

The address of a receive buffer is a function of the following:

- *Receive Global Base Address*. TDMxRGBA[RGBA], page 19-53.
- *Receive Channel Data Base Address*. TDMxRCPRn[RCDBA] field, page 19-61. RGBA << 16 + RCDBA points to the first byte of receive data buffer *n*. The four lsbs of RCDBA must be 0000.
- *Receive Data Buffer Displacement*. TDMxRDBDR[RDBD] field, page 19-66. Adding this field to the first byte of receive data buffer *n* indicates the location to which the TDM will write next: The address is calculated as follows: RGBA << 16 + RCDBA + RDBD. The RDBD can be used to show that data is written to the buffer and can be processed.

The address of a transmit buffer is a function of the following:

- *Transmit Global Base Address*. TDMxTGBA[TGBA] field, page 19-54.
- *Transmit Channel Data Base Address*. TDMxTCPRn[TCDBA] field, page 19-62. TGBA << 16 + TCDBA points to the first byte of transmit data buffer *n*. The four lsbs of TCDBA must be 0000.
- *Transmit Data Buffer Displacement*. TDMxTDBDR[TDBD] field, page 19-66. Adding this field to the first byte of transmit data buffer *n* indicates the location to which the TDM will read next: The address is calculated as follows: TGBA << 16 + TCDBA + TDBD. The TDBD can be used to show which data is already read from the buffer so that the buffer can be filled with new data.

Note: For A/μ-law channels the RDBD and the TDBD fields should be doubled before use.

Figure 19-25 illustrates the pointers associated with receive and transmit buffers that are mapped on the system I/F.

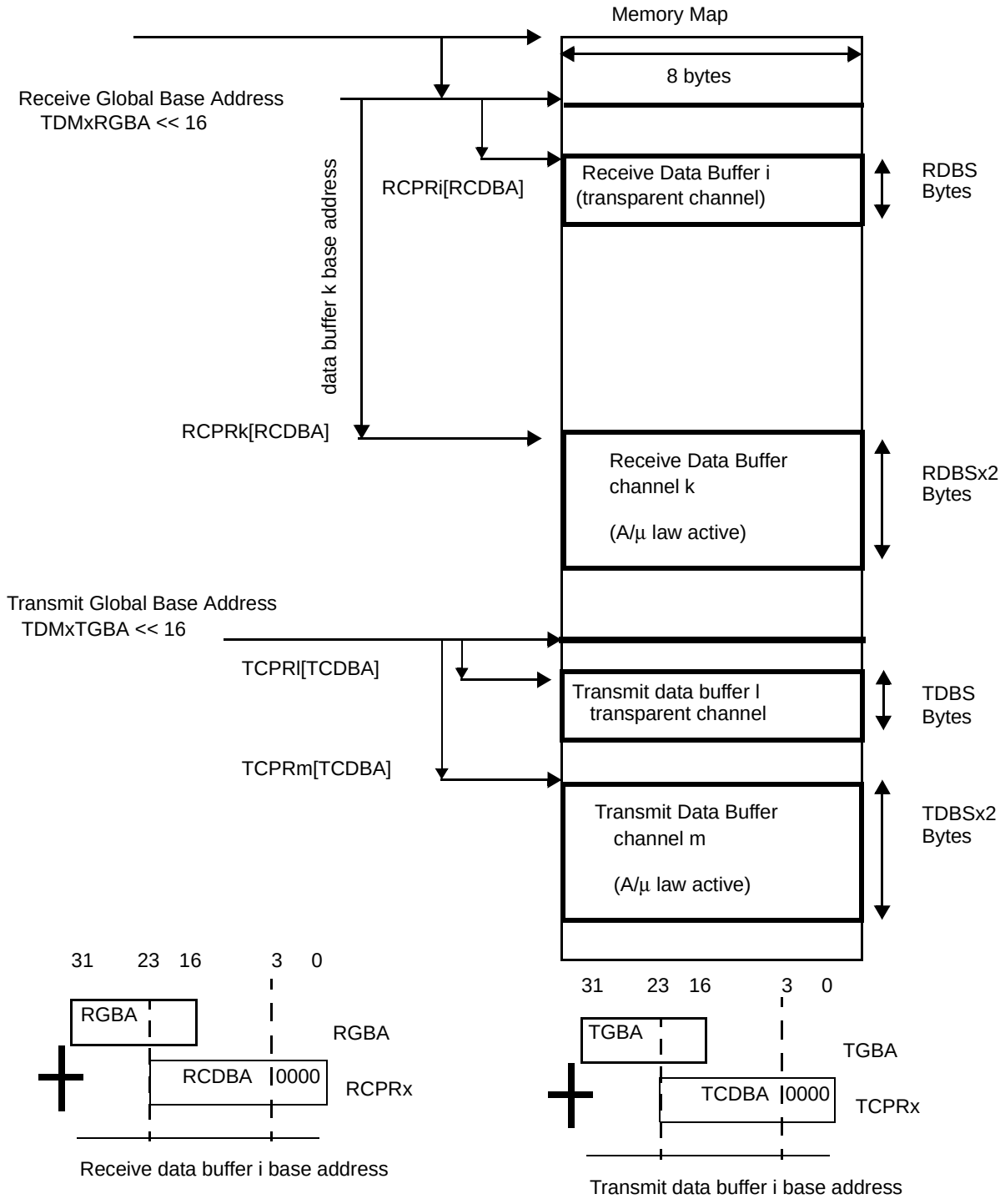


Figure 19-25. Data Buffer Location in Main Memory

19.2.6.3 Threshold Pointers and Interrupts

The receive data buffers share two threshold levels. The TDM notifies the SC3850 core each time it fills the receive buffer up to a threshold level. An example use of thresholds is the implementation of double buffering with the first threshold in the middle of a buffer and the second at the last eight bytes of the buffer.

When the TDM receiver fills the receive buffer through the system interface to an offset defined by the first threshold, which is the TDMxRDBFT[RDBFT] field (see page 19-58), the TDMxRER[RFTE] bit is set. If the TDMxRIER[RFTEE] bit is also set, a first threshold interrupt is generated. The interrupt can be generated as pulse or level, as determined by the TDMxRIR[RFTL] bit. If the interrupt is level, the ISR should clear the TDMxRER[RFTE] bit by writing a 1 to it. If the interrupt is pulse, then there is no need to clear the status bit. When the interrupt is asserted in the EPIC, then the SC3850 core can read all the receive buffers from their beginning up to the byte to which the first threshold (RDBFT) points. Meanwhile, the TDM keeps writing new data to the second part of the buffer.

When the TDM receiver fills the receive buffer through the system interface up to an offset defined by the second threshold, which is the TDMxRDBST[RDBST] field (see page 19-60), the TDMxRER[RSTE] bit is set. If the TDMxRIER[RSEEE] bit is also set, a second threshold interrupt is generated. The second threshold interrupt can generate as pulse or level, as determined by the TDMxRIR[RSTL] bit. If the interrupt is level, the ISR should clear the TDMxRER[RSTE] bit by writing a 1 to it. If the interrupt is pulse, there is no need to clear the status bit. When the interrupt is asserted in the EPIC, then the SC3850 core can read all the receive buffers up to the byte to which the second threshold (TDMxRDBST[RDBST]) points. Meanwhile, the TDM keeps writing new data to the first part of the buffer.

The transmit data buffers also share two threshold levels. The TDM notifies the SC3850 core each time it reads from the transmit buffer to a threshold level. When the TDM transmitter reads the transmit buffer through the system interface to an offset defined by the first threshold, which is the TDMxTDBFT[TDBFT] field, the TDMxTER[TFTE] bit is set. If the TDMxTIER[TFTEE] bit is also set, a first threshold interrupt is generated. The interrupt can generate as pulse or level, as determined by the TDMxTIR[TFTL] bit. If the interrupt is level, then the ISR should clear the TDMxTER[TFTE] bit by writing a 1 to it. If the interrupt is pulse, there is no need to clear the status bit. When the interrupt is asserted in the EPIC, then the SC3850 core can fill all the transmit buffers from their beginning up to the byte to which the first threshold (TDBFT) points. Meanwhile, the TDM continues reading new data from the second part of the buffer.

When the TDM transmitter reads the transmit buffer through the system interface up to an offset defined by the second threshold, which is the TDMxTDBST[TDBST] field, the TDMxTER[TSTE] bit is set. If the TDMxTEIR[TSTEE] bit is also set, a second threshold interrupt is generated. The second threshold interrupt can generate as pulse or level, as determined by the TDMxTIR[TSTL] bit. If the interrupt is level, the ISR should clear the TDMxTER[TSTE] bit by writing a 1 to it. If the interrupt is pulse, then there is no need to clear

the status bit. When the interrupt is asserted in the EPIC, then the SC3850 core can fill all the transmit buffers from their beginnings to the byte to which the second threshold (TDBST) points. Meanwhile, the TDM keeps reading new data from the buffer.

Figure 19-26 shows the threshold pointers for transparent and A/μ law channels.

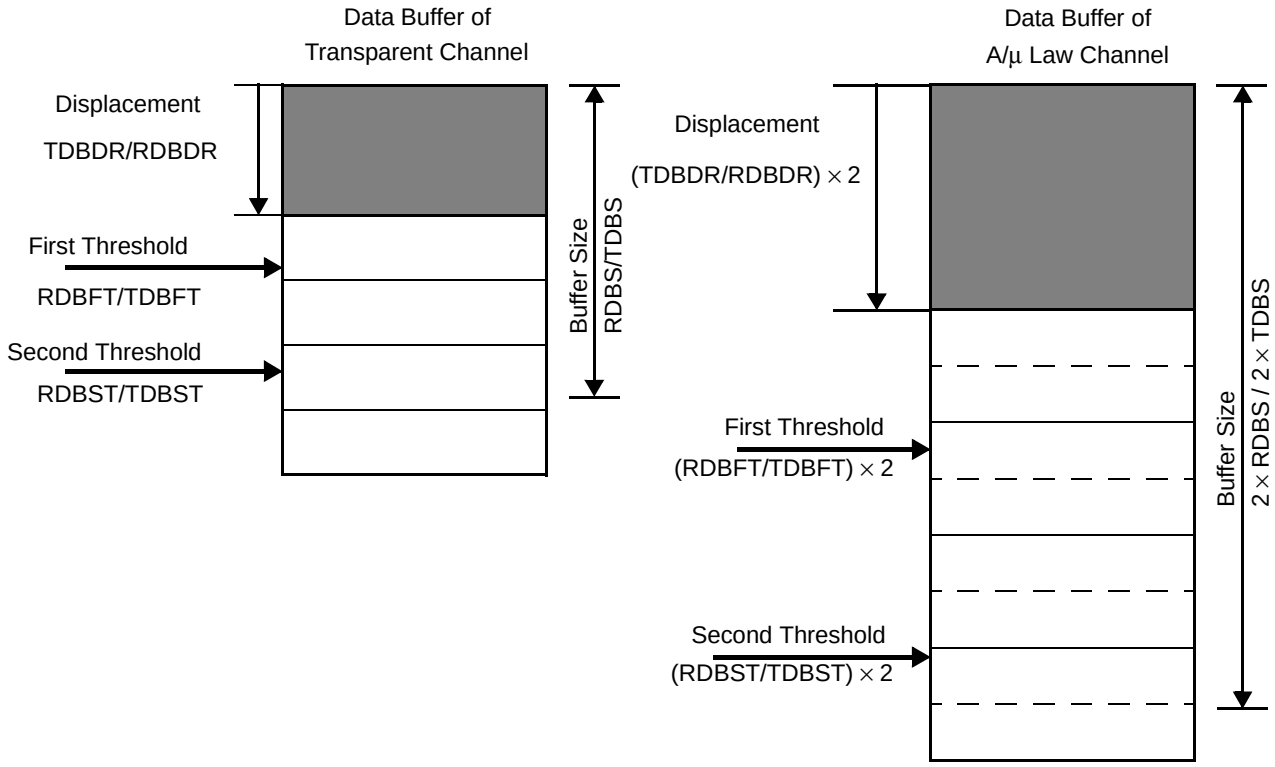


Figure 19-26. Main Memory Buffers Threshold Pointers

The TDMxRDBFT[RDBFT], TDMxRDBST[RDBST], TDMxTDBFT[TDBFT], and TDMxTDBST[TDBST] fields are control fields and can therefore be updated while the TDM is active. For example, to invoke an interrupt for each 64 bits written to the system I/F memory, the interrupt routine that handles the receive first threshold interrupt should include:

```
If (TDMxRDBFT[RDBFT] == (TDMxRDBS[RDBS] - 0xF))
    then TDMxRDBFT[RDBFT] = 0x0
else if (TDMxRDBFT[RDBFT] == (TDMxRDBS[RDBS] - 0x7))
    then TDMxRDBFT[RDBFT] = 0x8
else TDMxRDBFT[RDBFT] = TDMxRDBFT[RDBFT] + 0x10
```

The interrupt routine that handles the receive second threshold interrupt should include:

```
If (TDMxRDBST[RDBST] == (TDMxRDBS[RDBS] - 0xF))
    then TDMxRDBST[RDBST] = 0x0
else if (TDMxRDBST[RDBST] == (TDMxRDBS[RDBS] - 0x7))
    then TDMxRDBST[RDBST] = 0x8
else TDMxRDBST[RDBST] = TDMxRDBST[RDBST] + 0x10
```

19.2.6.4 Unified Buffer Mode

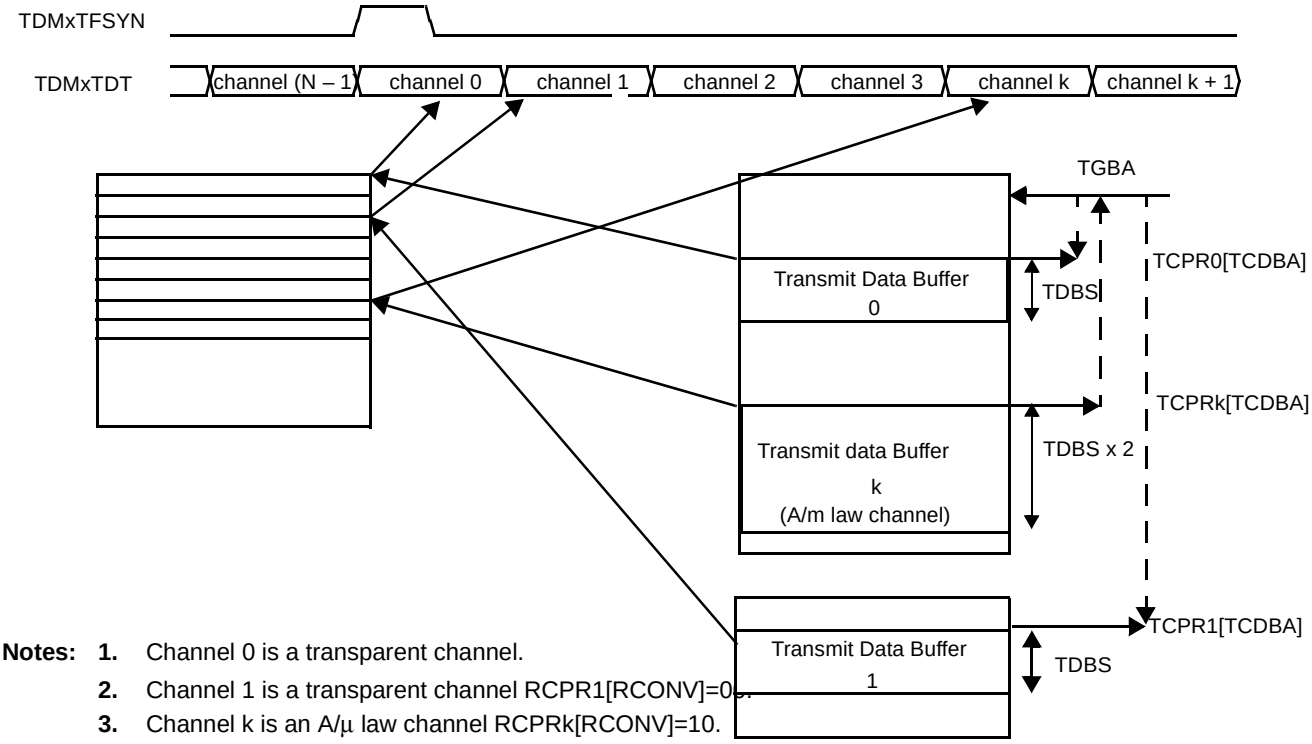
When the TDMxRFP[RUBM] bit is set (see page 19-47), all receive channels are directed to one buffer through the system interface. The number of active links must be one (RTSAL = 0b0000 or 0b0100 or 0b1100). The channel parameters of all the receive channels are located in the TDMxRCPR0 register. Unified Buffer mode essentially creates a one-channel link that is typically used in point-to-point connections. When TDMxTFP[TUBM] =1, data is transmitted from one buffer into all the transmit channels.

Note: When TDMxTFP[TUBM] = 1, the first block of data that was initialized for transmission in the memory is not transmitted. The size (number of bits) of the non-transmitted block is determined by the following equation:

$$2 \times (\text{TDMxTFP[TNCF]} - 1) \times (\text{TDMxTFP[TCS]} + 1)$$

For example, if the transmit number of channels (that is, TDMxTFP[TNCF] + 1) is 32 and the transmit channel size (that is, TDMxTFP[TCS] + 1) is 8 bits, then the number of non transmitted bits is 480. Note that those bits (that are not be transmitted) are located either in the TDM local memory and/or in the device level memory (M2, M3, DDR, and so on).

Figure 19-27 describes the transmit data flow in independent data buffers mode (TDMxTFP[TUBM] = 0). Each data channel transfers data from an independent data buffer to the TDM local memory, and transmit data is read out serially from the local memory.



Notes:
1. Channel 0 is a transparent channel.
2. Channel 1 is a transparent channel RCPR1[RCONV]=0.
3. Channel k is an A/μ law channel RCPRk[RCONV]=10.

Figure 19-27. Transmit Data Buffer in Independent Data Buffer Mode (TUBM=0)

Figure 19-28 illustrates the receive data flow in receive unified buffer mode. All the received channels are stored in the TDM local memory and then written into their unified buffer through the system I/F.

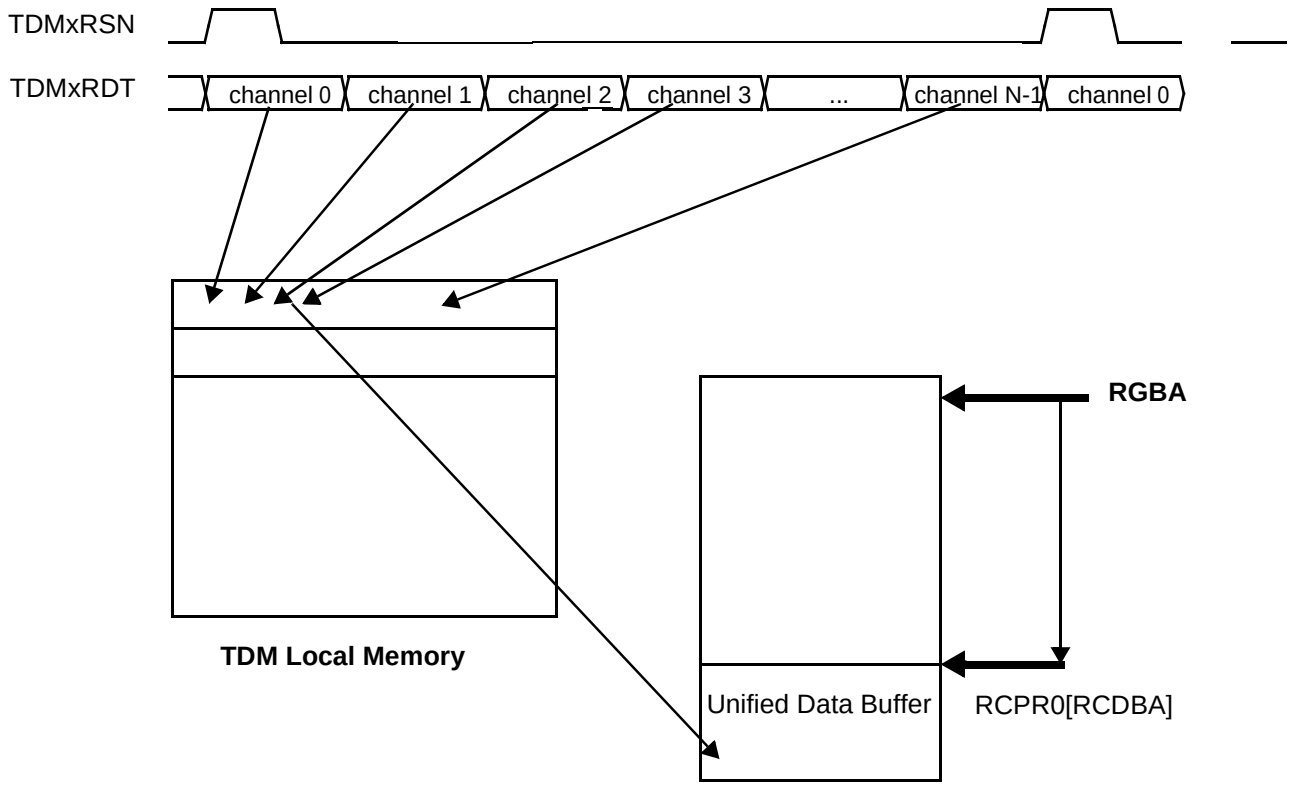


Figure 19-28. Receive Unified Buffer Mode (RUBM = 1)

Note: When the receiver is configured as Unified Buffer Mode, the RRDO bit in the TDMxRIR should be cleared. When the transmitter is configured as Unified Buffer Mode, the TRDO bit in the TDMxTIR should be cleared.

19.2.7 Adaptation Machine

Each TDM module has an Adaptation Machine that counts the number of bits between frame SYNCs. This module can be used to determine the frame size in bits.

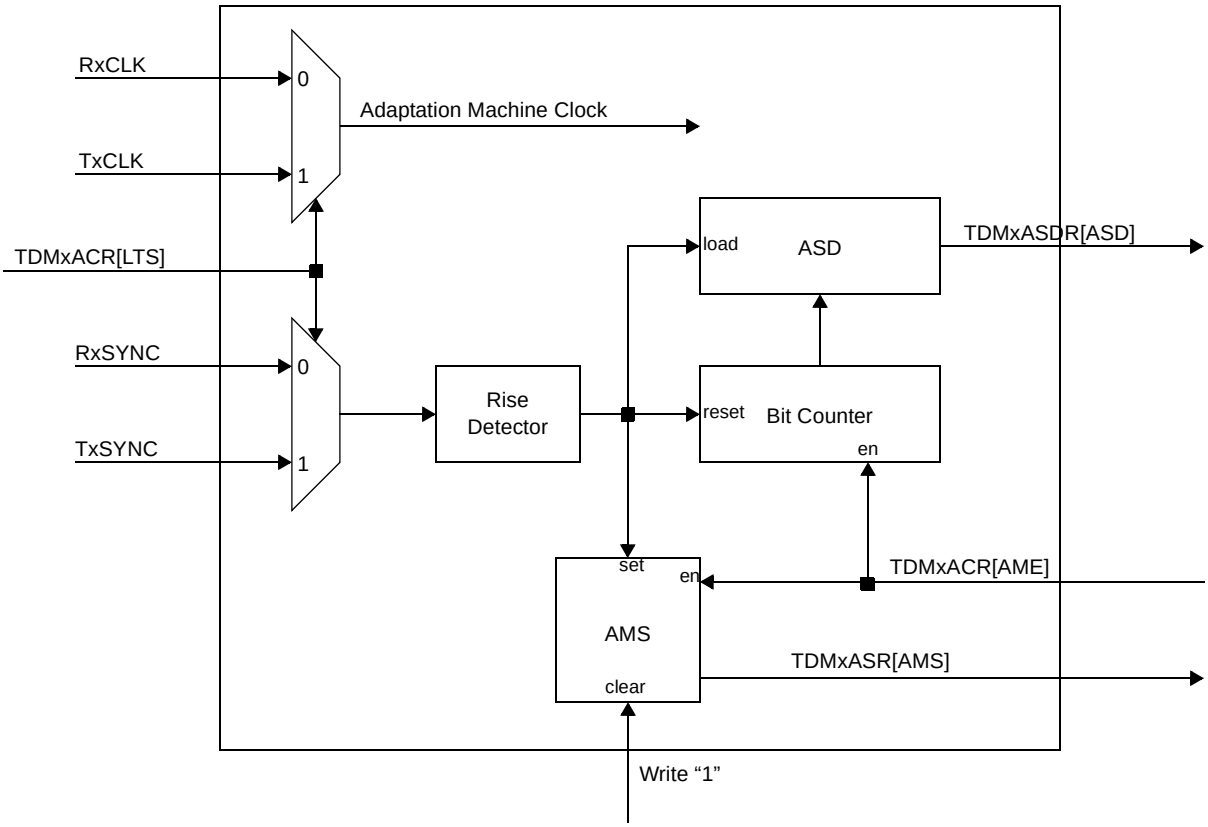


Figure 19-29. Adaptation Machine Block Diagram

Figure 19-29 shows the Adaptation Machine block diagram. The Adaptation Machine can work with either the transmit or receive frame. The LTS bit in the TDMx Adaptation Control Register (TDMxACR) defines whether the Adaptation Machine is fed with the transmit or with the receive frame sync and clock. The Adaptation Machine samples the sync only at the rising edge of the associated clock. When enabled, the Adaptation Machine detects a frame sync, stores the Bit Counter in the ASD field in the TDMx Adaptation Sync Distance Register (TDMxASDR), resets the Bit Counter, and sets the AMS bit in the TDMx Adaptation Status Register (TDMxASR).

The following steps define how to use the Adaptation Machine:

1. Configure the LTS bit to define whether the Adaptation Machine is fed with the Transmit or with the receive frame sync and clock. (See page 19-56).
2. Set the AME bit in the TDMxACR to enable the Adaptation Machine.
3. Wait for AMS bit in the TDMxASR to be set to 1. (See the TDMx Adaptation Status Register on page 19-70).

4. Read the value of the ASD field in the TDMxASDR. (See the TDMxASDR indicates the number of receive/transmit bits between the last two consecutive receive/transmit sync events. The register value updates each time the TDMxASR[AMS] bit is set. on page 19-70).
5. Clear the AMS bit by writing a 1 to the AMS bit in the TDMxASR.
6. Repeat steps 3–5 until you read the same value from the ASD field for 20 consecutive times. At this time the ASD value is valid and can be used to configure the TDM receiver or transmitter.
7. Clear the AME bit in the TDMxACR to disable the Adaptation Machine.
8. Configure the receiver or transmitter according to the following parameters:
   - ASD value.
   - Number of active links.
   - Channel size.
   - SYN

19.3 TDM Power Saving

The MSC8156E TDMs use the stop mode of different clocks to save power. Each TDM has three clock domains: transmit serial, receive serial, and the system clock (CLASS clock/2). The transmit serial clock is not supplied to the TDM module when the transmitter is disabled, that is, the TDMxTCR[TEN] bit and the TDMxTSR[TENS] are both clear. The receive serial clock is not supplied to the TDM module when the receiver is disabled, TDMxRCR[REN] bit and TDMxRSR[RENS] bit are both clear. The system clock automatically stops when the TDM is disabled, that is, both transmitter and receiver are disabled. In addition, the TDM registers get the system clock only at reset or during an access.

19.4 Channel Activation

The TACT and RACT bits in the Transmit/Receive Channel Parameter n Registers (see page 19-61 and page 19-62) are enabled during the receiver/transmitter operation to control the channel activation. If the TACT/RACT bit is clear, the channel is not active. Otherwise, it is active. The procedure for activating an inactive receive channel (C) is as follows:

1. Verify that the active (RACT) bit of the channel is clear.
2. Write the initialization value to the channel locations in the receive TDM local memory.

   The receive local memory contains 1, 2, 4, 8, 16, or 32 buffers so that each buffer contains 8 bytes per channel. The location of channel C in buffer B is the 8 bytes that start at $(256 / (RNB + 1) \times B + C) \times 8$. (See **Section 19.2.3**, *TDM Data Structures*, on page 19-13). For example, if the number of buffers is four, the SC3850 core should write the

initialization value to all four receive buffers. Initializing the receive TDM local memory prevents invalid data from being received by the channel buffer in the main memory.

3. Set the TDMxRCPRC[RACT] bit (C indicates the channel number).

The procedure for activating an inactive transmit channel (C) is as follows:

1. Verify that the active (TACT) bit of the channel is clear.
2. Write the initialization value to the channel locations in the transmit TDM local memory.

   The transmit local memory contains 1, 2, 4, 8, 16, or 32 buffers so that each buffer contains 8 bytes per channel. The location of channel C in buffer B is the 8 bytes that start at (256 / (TNB + 1) × B + C) × 8. (See **Section 19.2.3**, *TDM Data Structures*, on page 19-13). Initializing the transmit TDM local memory prevents invalid data from being transmitted out.
3. Set the TDMxTCPRC[TACT] bit (C indicates the channel number).

   For example, if the SC3850 core needs to activate receive channel 2 and the number of receive buffers is 4 (RNB[3–0] = 0011), it should write the initialization value to the following addresses (which are offsets from the TDMx receive local memory, see **Chapter 9**, *Memory Map*):
   - 0x0010–0x0017 (the channel location in buffer 0).
   - 0x0210–0x0217 (the channel location in buffer 1).
   - 0x0410–0x0417 (the channel location in buffer 2).
   - 0x0610–0x0617 (the channel location in buffer 3).

19.5 Loopback Support

In Loopback Test mode, the receiver receives the same data that is transmitted. The frame clock should supply to the TDM, and the frame sync can be generated internally or supplied externally. The receiver and transmitter share the frame sync, frame clock, and data links (RTSAL[3–2] = 0b11). The number of data links can be 1, 2, or 4 and is determined by the RTSAL[1–0] bits. All the receive and transmit frame channels are active. The procedure for loopback is as follows:

1. Configure the RTSAL field in the GIR register (see page 19-36) to shared data links mode- RTSAL[3–2] = 0b11. The number of data links can be 1, 2, or 4.
2. Configure the receive and transmit frame parameters to be the same. The configuration of the RFP register should be identical to that of the TFP register (see page 19-47 and page 19-50).
3. Configure the TDMx Transmit Interface Register (TDMxTIR) and the TDMx Receive Interface Register (TDMxRIR) according to the following instructions:
   - Set the Transmit Sync Out (TSO) bit to 1. The transmit sync is generated by the TDM.

- Set the Receive Frame Sync Delay field to 0x00 and the Transmit Frame Sync Delay field to 0x01.
- Set both the Receive Frame Sync Edge (RFSE) bit and the Transmit Frame Sync Edge (TFSE) bits to 1. The sync samples at the negative edge.
- The value of the Receive Sync level bit should be identical to that of the Transmit Sync Level field (RSL = TSL).
- Clear the Receive Data Edge bit (RDE = 0x0), so that the receive data is sampled on the positive edge.
- Set the Transmit Data Edge bit (TDE = 0x1) to transmit data driven at the negative edge.

4. Set the receive active RACT bit of all the channels to 1.
5. Set the transmit active TACT bit of all the channels to 1.
6. Set the TDMxTCR[TEN] bit.
7. Set the TDMxRCR[REN] bit.

19.6 TDM Initialization

After reset, all TDM registers are reset. **Table 19-5** describes the TDM signal direction after reset.

Table 19-5. TDM Signal Direction at Reset

TDM Signal	Signal Direction
TDMXRCK	input
TDMxRDT	input
TDMxRSN	input
TDMxTCK	input
TDMxTDT	input
TDMxTSN	input

The TDMxRCR[REN] bit (see page 19-57) enables the receiver part of the TDM module. When TDMxRCR[REN] is clear, the receive TDM is disabled, but all the registers retain their values except for the TDMx Receive Data Buffers Displacement Register (TDMxRDBDR). The TDMxTCR[TEN] bit (see page 19-58) enables the transmit part of the TDM module. When TDMxTCR[TEN] is clear, the transmit TDM is disabled, but all the registers retain their values except for the TDMx Transmit Data Buffers Displacement Register (TDMxTDBDR).

The correct flow for initializing the TDM is as follows:

1. Perform a hardware reset to disable the receiver and the transmitter.
2. Program all the configuration registers. Program all the control registers, except the TDMx Receive Control Register (TDMxRCR) and the TDMx Transmit Control Register (TDMxTCR).

3. Fill the sync data in all the TDM receive local memory.

   The received data is stored in 256 entries of 8 bytes each located in the addresses between 0x0000–0x07FF. This memory contains 1, 2, 4, 8, 16, or 32 indexed buffers, starting at 0. Each buffer contains multiple frames. The number of buffers used to store the received data is indicated in the RNB field of the TDMx Receive number of Buffers Register (TDMxRNB) (see page 19-67). Channel C in buffer B is the 8 bytes starting at $(256 / (RNB + 1) \times B + C) \times 8$. (Refer to **Section 19.2.3**, *TDM Data Structures*, on page 19-13 for details)

4. Fill the sync data in all the TDM transmit local memory.

   Transmit data is located in the TDM local memory before it is transmitted externally. The data is stored in 256 8-byte entries in addresses between 0x1800–0x27FF. This memory can contain 1, 2, 4, 8, 16, or 32 indexed buffers starting at 0. Each buffer contains multiple frames. The number of buffers used to store the transmitted data is indicated in the TNB field of the TDMx Transmitter Number of Buffers Register (TDMxTNB). Channel C in buffer B is the 8 bytes starting at $(256/(TNB+1) \times B+C) \times 8$.

5. Clear the TDMxRER and TDMxTER event registers by writing a value of 0xF to each of them.

6. Set the TDMxRCR[REN] bit and/or the TDMxTCR[TEN] bit.

19.7 TDM Programming Model

The handshake between the TDM module and the SC3850 core occurs via a set of registers, data structures in the memory, and interrupts. All TDM registers are mapped into the CCSR address space. See **Chapter 9**, *Memory Map* for details on CCSR addressing. There are four modules (TDM 0–3), each with its own region in the CCSR address space. Within the module address space, the area is divided into spaces for configuration registers, control registers, and status registers as follows:

- *Configuration registers*. Set the operation modes and provide indications for all channels. They are set before the TDM is enabled and should not be changed while the TDM is active.
- *Control registers*. Set the channel specific parameters individually for each channel and the threshold pointers. These registers can be changed during operation.
- *Status registers*. Read-only registers that can be accessed any time.

This section describes the TDM module registers, which are listed as follows:

Note: The base addresses for the TDM registers are as follows:

- TDM0 = 0xFFF30000
- TDM1 = 0xFFF34000
- TDM2 = 0xFFF38000
- TDM3 = 0xFFF3C000

19.7.1 Configuration Registers

The following sections describe the configuration registers.

19.7.1.1 TDMx General Interface Register (TDMxGIR)

TDM*x*GIR TDM*x* General Interface Register Offset 0x3FF8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—										SYNC_ MODE	CTS	RTSAL			
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*GIR defines the TDM*x* interface operation mode.

Table 19-6. TDM*x*GIR Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to zero for future compatibility.	
SYNC_MODE 5	0	**Sync Mode** Used to synchronize the start of each TDM receiver/transmitter with enabling the TDM0 transmitter. In this mode, the TDM0 transmitter should be the last to operate.	0 Non-sync mode. 1 Sync mode.
CTS 4	0	**Common TDM Signals** Defines whether the TDM shares sync and clock signals with other TDM modules. The four TDMs can share signals as follows: ■ TDM[0–3] do not share signals. (TDMxCTS = 0 for 0 ≤ x ≤ 3) ■ TDM0 and TDM1 share signals, but TDM[2–3] do not share signals with the other TDM modules. (TDMxCTS = 1for 0 ≤ x ≤ 1, TDMyCTS=0 for 2 ≤ y ≤ 3) ■ TDM[0–2] share signals, but TDM[3] does not share signals with the other TDM modules. (TDMxCTS=1for 0 ≤ x ≤ 2, TDMyCTS=0 for 3 = y) ■ TDM[0–3] share signals (TDMxCTS=1for 0 ≤ x ≤ 3) **Table 19-7** on page 19-38 describes the functionality of the TDM signals as a function of the **RTSAL** field. **Note:** If the TDM modules share sync and clock signals, then the RFP, TFP,RIR, and TIR registers should be configured the same way for all the TDM modules.	0 The TDM does not share signals with other TDM modules 1 The TDM shares sync and clock signals with other TDM modules. Refer to **Table 19-7**.

Table 19-6. TDM*x*GIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RTSAL 3–0	0	**Receive and Transmit Sharing and Active Links** Defines the TDM serial interface operating mode. It determines whether the TDM transmit and receive paths are independent or share the same clock and sync. It also determines whether the TDM receive and transmit share the data links. Bits 2 and 3 determine the receive and transmit sharing mode, and bits 1 and 0 determine the number of active data links. **Note:** If RTSAL [3–2]= 01 or 11, some parameters of the receive and transmit path should be the same. The value of the TDMxRFP[RNCF], RCS and RT1 fields should be equal to that of the TDMxTFP[TNCF], TCS, and TT1 fields. The value of the TDMxRIR[RFSE] and TDMxRIR[RSL] fields should be equal to the that of the TDMxTIR[TFSE] and TDMxTIR[TSL] fields, respectively. For details, see **Section 19.2.1**, *Common Signals for the TDM Modules,* on page 19-8. **Note:** Unused signals should not be configured as dedicated signals in the PAR.	0000 The receive and transmit are independent.The TDM receives one data link and transmits one data link. 0001 The receive and transmit are independent. The TDM receives two data links and transmits two data links (valid only if CTS=1). 0010 Reserved 0011 Reserved. 0100 The receive and transmit share the frame clock and frame sync.The TDM receives one data link and transmits one data link. 0101 The receive and transmit share the frame sync and frame clock. The TDM receives two data links and transmits two data links. 0110–1011 Reserved. 1100 The receive and transmit share the frame sync, frame clock, and one full duplex data link. 1101 The receive and transmit share the frame sync, frame clock, and two full duplex data links. 1110 Reserved. 1111 The receive and transmit share the frame sync, frame clock, and four full duplex data links. Refer to **Table 19-8**

Table 19-7. TDM Signal Configuration When TDM Modules Share Signals

RTSAL[3–0] Field Value	Description
0000	Receive clock: TDM1TCK Transmit clock: TDM0TCK Receive sync: TDM1TSN Transmit sync: TDM0TSN Receive data links: TDMxRDT Transmit data links: TDMxTDT Unused signals: TDMxRCK, TDMxRSN. **Note:** The x specifies the TDM number of TDMs that share signals. For example if TDM0, TDM1, and TDM2 share signals, then x is equal to 0,1, and 2 and the receive data links are TDM0RDT, TDM1RDT, and TDM2RDT.
0001	Receive clock: TDM1TCK Transmit clock: TDM0TCK Receive sync: TDM1TSN Transmit sync: TDM0TSN Receive data links: TDMxRDT, TDMxRSN. Transmit data links: TDMxTDT, TDMxRCLk. **Note:** The x specifies the number of the TDM and any one of the shared TDM modules.
0100	Frame clock (receive and transmit share the same clock): TDM0TCK Frame sync (receive and transmit share the same sync): TDM0TSN Receive data links: TDMxRDT Transmit data links: TDMxTDT Unused signals: TDMxRCK, TDMxRSN, TDMyTCK, TDMyTSN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM0RCK, TDM1RCK, TDM0RSN, TDM1RSN, TDM1TCK, and TDM1TSN.
0101	Frame clock (receive and transmit share the same clock): TDM0TCK Frame sync (receive and transmit share the same sync): TDM0TSN Receive data links: TDMxRDT, TDMxRSN Transmit data links: TDMxTDT, TDMxRCK Unused signals: TDMyTCK, TDMyTSN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM1TCK and TDM1TSN.
1100	Frame clock (receive and transmit share the same clock): TDM0TCK Frame sync (receive and transmit share the same sync): TDM0TSN Full duplex data links (the data link is inout and is used for receive and transmit) TDMxRDT. Unused signals: TDMyTCK, TDMyTSN, TDMxRCK, TDMxRSN, and TDMxTDT. TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM0RCK, TDM1RCK, TDM0RSN, TDM1RSN, TDM0TDT, TDM1TDT, TDM1TCK, and TDM1TSN.
1101	Frame clock (receive and transmit share the same clock): TDM0TCK Frame sync (receive and transmit share the same sync): TDM0TSN Full duplex data links (the data link is inout it and it is used for receive and transmit): TDMxRDT, TDMxRSN Unused signals: TDMyTCK, TDMyTSN, TDMxRCK, TDMxTDT. TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except of TDM0. for example if TDM0 and TDM1 shared signals then the unused signals are TDM0RCK, TDM1RCK, TDM0TDT, TDM1TDT, TDM1TCK, and TDM1TSN.

Table 19-7. TDM Signal Configuration When TDM Modules Share Signals (Continued)

RTSAL[3–0] Field Value	Description
1111	Frame clock (receive and transmit share the same clock): TDM0TCK Frame sync (receive and transmit share the same sync): TDM0TSN Full duplex data links (the data link is inout and is used for receive and transmit): TDMxRDT,TDMxRSN,TDMxTDT,TDMxRCK Unused signals: TDMyTCK, TDMyTSN TDMx specifies the TDM number and any one of the shared TDM modules. TDMy specifies the TDM number and any one of the shared TDM modules except TDM0. For example, if TDM0 and TDM1 share signals, the unused signals are TDM1TCK and TDM1TSN.

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields

No.	CTS	RTSAL [3–0]	TDMxRDT	TDMxRSN	TDMxRCK	TDMxTDT	TDMxTSN	TDMxTCK	Comments
0	0	0000	receive data (RDATA_A)	receive sync	receive clock	transmit data (TDATA_A)	transmit sync	transmit clock	The TDM does not share signals with others TDM modules. Independent mode. One active data link.
	direction		Input	Input	Input	Output	Inout	Input	
1	0	0001	Reserved						
2	0	0010	Reserved						
3	0	0011	Reserved						
4	0	0100	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	frame sync	frame clock	The TDM does not share signals with others TDM modules. Receive and transmit share sync and clock signals. One active data link.
	direction		input			Output	Inout	Input	

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDT	TDMxRSN	TDMxRCK	TDMxTDT	TDMxTSN	TDMxTCK	Comments
5	0	0101	receive data (RDATA_A)	receive data (RDATA_B)	transmit data (TDATA_B)	transmit data (TDATA_A)	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share sync and clock signals. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
6	0	0110	Reserved						
7	0	0111	Reserved						
8	0	1100	data link (DATA_A)	not used	not used	not used	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. One full duplex active data link.
	direction		Inout				Inout	Input	
9	0	1101	data link (DATA_A)	data link (DATA_B)	not used	not used	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. Two full duplex active data links.
	direction		Inout	Inout			Inout	Input	
10	0	1110	Reserved						

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDT	TDMxRSN	TDMxRCK	TDMxTDT	TDMxTSN	TDMxTCK	Comments
11	0	1111	data link (DATA_A)	data link (DATA_B)	data link (DATA_D)	data link (DATA_C)	frame sync	frame clock	The TDM does not share signals with other TDM modules. Receive and transmit share the sync, clock, and data signals. Four full duplex active data links.
	direction		Inout	Inout	Inout	Inout	Inout	Input	
12	1	0000	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	receive sync/ transmit sync/ not used	receive clock/ transmit clock/ not used	The TDM shares receive sync and clock and transmit sync and clock with other TDM modules. Independent mode. One active data link.
	direction		Input			Output	Inout	Input	
13	1	0001	receive data (RDATA_A)	receive data (RDATA_B)	transmit data (TDATA_B)	transmit data (TDATA_A)	receive sync/ transmit sync/ not used	receive clock/ transmit clock/ not used	The TDM shares receive sync and clock and transmit sync and clock with other TDM modules. Independent mode. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
14	1	0010	Reserved						
15	1	0011	Reserved						
16	1	0100	receive data (RDATA_A)	not used	not used	transmit data (TDATA_A)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit shared sync and clock signals. One active data link.
	direction		input			Output	Inout	Input	

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDT	TDMxRSN	TDMxRCK	TDMxTDT	TDMxTSN	TDMxTCK	Comments
17	1	0101	receive data RDATA_A	receive data RDATA_B	transmit data (TDATA_B)	transmit data (TDATA_A)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync and clock signals. Two active data links.
	direction		Input	Input	Output	Output	Inout	Input	
18	10	0110	Reserved						
19	1	0111	Reserved						
20	1	1100	data link (DATA_A)	not used	not used	not used	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. One full duplex active data link.
	direction		Inout				Inout	Input	
21	1	1101	data link (DATA_A)	data link (DATA_B)	not used	not used	frame sync/ not used	frame clock/ not used	The TDM share the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. Two full duplex active data links.
	direction		Inout	Inout			Inout	Input	
22	1	1110	Reserved						

Table 19-8. TDM Signal Configuration as a Function of the RTSAL and CTS Fields (Continued)

No.	CTS	RTSAL [3–0]	TDMxRDT	TDMxRSN	TDMxRCK	TDMxTDT	TDMxTSN	TDMxTCK	Comments
23	1	1111	data link (DATA_A)	data link (DATA_B)	data link (DATA_D)	data link (DATA_C)	frame sync/ not used	frame clock/ not used	The TDM shares the frame sync and frame clock with other TDM modules. Receive and transmit share the sync, clock, and data signals. Four full duplex active data links.
	direction		Inout	Inout	Inout	Inout	Inout	Input	
Note: Frame sync specifies that the receiver and transmitter use the same sync. Frame clock specifies that the receiver and transmitter use the same clock. If data link specifies that the direction is inout, the signal is used for receive and transmit.									

19.7.1.2 TDMx Receive Interface Register (TDMxRIR)

TDM*x*RIR TDM*x* Receive Interface Register Offset 0x3FF0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															RBOR
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Boot	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
	RFTL	RSTL	—								RFSD		RSL	RDE	RFSE	RRDO
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RIR defines the TDM*x* receiver interface operation.

Table 19-9. TDM*x*RIR Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
RBOR 16	1	**Receive Byte Order** Indicates the location of the receive data in the receive external memory buffer. The boot program writes a 1 to this bit.	0 First receive data is at the high address. 1 First receive data is at the low address.
RFTL 15	0	**Receive First Threshold Level** Determines whether the receive first threshold interrupt is pulse or level. For details, see **Section 19.2.6.3**.	0 Receive first threshold interrupt is pulse. 1 Receive first threshold interrupt is level.

Table 19-9. TDM*x*RIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RSTL 14	0	**Receive Second Threshold Level** Determines whether the receive second threshold interrupt is pulse or level. For details, see **Section 19.2.6.3**.	0 Receive second threshold interrupt is pulse. 1 Receive second threshold interrupt is level.
— 13–6	0	Reserved. Write to zero for future compatibility.	
RFSD 5–4	0	**Receive Frame Sync Delay** With the RDE and the RFSE bits, determines the number of clocks between the receive sync signal and the first data bit of the receive frame.	Refer to **Table 19-10**. **Note:** If the receive channel size is 2 (RCS = 0x1), then the RFSD field value can be only 0 or 1.
RSL 3	0	**Receive Sync Level** Determines the polarity of the receive sync signal. For details, see **Figure 19-21**.	0 Receive sync is active on logic 1. 1 Receive sync is active on logic 0.
RDE 2	0	**Receive Data Edge** Determines whether the receive data signal is sampled on the rising or falling edge of the receive clock. For details see **Section 19.2.4.2**.	0 The receive data signal is sampled on the rising edge of the receive clock. 1 The receive data signal is sampled on the falling edge of the receive clock.
RFSE 1	0	**Receive Frame Sync Edge** Determines whether the receive frame sync signal is sampled on the rising or falling edge of the receive clock. For details, see **Section 19.2.4.2**.	0 The receive frame sync signal is sampled with the rising edge of the receive clock. 1 The receive frame sync signal is sampled on the falling edge of the receive clock.
RRDO 0	0	**Receive Reversed Data Order** For examples, see **Section 19.2.4.2**.	0 The first bit of a received channel is stored as the most significant bit in the internal memory. 1 The first bit of a received channel is stored as the least significant bit in the internal memory

Table 19-10. Received Data Delay for Receive Frame Sync

Frame Sync Delay	Frame Sync Edge	Data Edge	Receive Clocks[1]
00	0	0	0.0
00	0	1	0.5
00	1	0	0.5
00	1	1	0.0
01	0	0	1.0
01	0	1	1.5
01	1	0	1.5
01	1	1	1.0
10	0	0	2.0
10	0	1	2.5
10	1	0	2.5
10	1	1	2.0
11	0	0	3.0
11	0	1	3.5
11	1	0	3.5
11	1	1	3.0

Note: Receive clocks is the number of receive clocks between the sample of the receive frame sync and the sample of first data bit of the received frame.

19.7.1.3 TDMx Transmit Interface Register (TDMxTIR)

TDM*x*TIR TDM*x* Transmit Interface Register Offset 0x3FE8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															TBOR
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Boot	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
	TFTL	TSTL	TSO	TAO	SOL	SOE	—				TFSD		TSL	TDE	TFSE	TRDO
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxTIR defines the TDM *x* transmitter interface operation.

Table 19-11. TDM*x*TIR Bit Descriptions

Name	Reset	Description	Settings
— 31–17	0	Reserved. Write to zero for future compatibility.	
TBOR 16	1	**Transmit Byte Order** Indicates how to transmit data residing in the transmit external memory buffer. The boot program writes a 1 to this bit.	0 Transmit high address first. 1 Transmit low address first.

Table 19-11. TDM*x*TIR Bit Descriptions (Continued)

Name	Reset	Description	Settings
TFTL 15	0	**Transmit First Threshold Level** Determines whether the Transmit first threshold interrupt is pulse or level. For details, see **Section 19.2.6.3.**	0 Transmit first threshold interrupt is pulse. 1 Transmit first threshold interrupt is level.
TSTL 14	0	**Transmit Second Threshold Level** Determines whether the Transmit second threshold interrupt is pulse or level. For details, see **Section 19.2.6.3**.	0 Transmit second threshold interrupt is pulse. 1 Transmit second threshold interrupt is level.
TSO 13	0	**Transmit Sync Output** Determines whether the transmit sync is driven out by the TDM transmitter or it input to the TDM transmitter. For details, see **Section 19.2.4.1**	0 Transmit sync is input. 1 Transmit sync is output.
TAO 12	0	**Transmit Always Output** Determines whether the TDM transmitter drives TDMxDAT for the inactive channels.	0 The TDM transmitter does not drive the TDMxDAT for inactive channels. 1 The TDM transmitter drives the TDMxDAT regardless of whether the channel is active.
SOL 11	0	**Sync Out Length** Indicates whether the TDMxTSN is asserted for one cycle of TDMxTCK or is asserted for the duration of the first channel in the frame. For details, see **Section 19.2.4.1**.	0 The sync_out width is one bit. 1 The sync_out width is equal to the channel width.
SOE 10	0	**Sync Out Edge** Determines whether the sync out signal is driven out with the rising or falling edge of the transmit clock. For details, see **Section 19.2.4.1**.	0 The transmit frame sync signal is driven out on the rising edge of the transmit clock. 1 The transmit frame sync signal is driven out on the falling edge of the transmit clock.
— 9–6	0	Reserved. Write to zero for future compatibility.	
TFSD 5–4	0	**Transmit Frame Sync Delay** With the TDE and the TFSE bits, determines the number of clocks between the transmit sync signal and the first data bit of the transmit frame. For examples, see **Section 19.2.4.2**.	Refer to Table 19-12 on page 47. **Note:** If the transmit channel size is 2 (TCS = 0x1) then the TFSD field value can be only 0 or 1.
TSL 3	0	**Transmit Sync Level** Determines the polarity of the transmit sync signal. For details, see **Section 19.2.4.2**.	0 Transmit sync is active on logic 1. 1 Transmit sync is active on logic 0.
TDE 2	0	**Transmit Data Edge** Determines whether the transmit data is driven out on the rising or falling edge of the transmit clock. For details, see **Section 19.2.4.2**.	0 The transmit data is driven out on the rising edge of the transmit clock. 1 The transmit data is driven out on the falling edge of the transmit clock.
TFSE 1	0	**Transmit Frame Sync Edge** Determines whether the transmit frame sync signal is sampled with the rising or falling edge of the transmit clock. For details, see **Section 19.2.4.2**.	0 The transmit frame sync signal is sampled with the rising edge of the transmit clock. 1 The transmit frame sync signal is sampled with the falling edge of the transmit clock.
TRDO 0	0	**Transmit Reversed Data Order** For examples, see **Section 19.2.4.4**.	0 The most significant bit of the memory is sent out at the first transmit data bit. 1 The least significant bit of the memory is sent out at the first transmit data bit.

Table 19-12. Transmit Data Delay for Transmit Frame Sync

Frame Sync Delay	Frame Sync Edge	Data Edge	Transmit Clocks[1]
00	0	0	–1 [2]
00	0	1	–0.5 [2]
00	1	0	–0.5 [2]
00	1	1	–1 [2]
01	0	0	0
01	0	1	0.5
01	1	0	0.5
01	1	1	0
10	0	0	1
10	0	1	1.5
10	1	0	1.5
10	1	1	1
11	0	0	2
11	0	1	2.5
11	1	0	2.5
11	1	1	2

Notes: 1. Transmit clocks is the number of transmit clocks between the first transmit frame sync sample and the first data bit of the frame that is driven out.

2. The field value is negative because the data is driven out before the transmit frame sync sample.

19.7.1.4 TDMx Receive Frame Parameters (TDMxRFP)

TDM*x*RFP TDM*x* Receive Frame Parameters Offset 0x3FE0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								RNCF							
Type	R/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
	—					RCDBL			—		RCS				RT1	RUBM
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RFP defines the TDM*x* receive frame parameters.

Table 19-13. TDM*x*RFP Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RNCF 23–16	0	**Receive Number of Channels in a TDM Frame** Specifies the total number of channels that are received in the TDM modules. One TDM frame can contain 2–256 channels at a granularity of two. **Notes:** **1.** RNCF[8-15] = (number of channels that received on one active link) × (number of active data links) – 1, the number of active data links is specified in the RTSAL field. **2.** If RCDBL field is clear, then the minimum number of channels is limit. The minimum receive number of channels is 128 / (receive channel size) + 2. For example, if the channel size is 4 bits, then the receive TDM frame should contain at least 34 channels. **Table 19-14** describes the RNCF valid value as a function of the RTSAL field (Receive and Transmit Sharing and Active Links). For details, see **Section 19.2.5**.	0x00 Reserved. 0x01 2 received channels. 0x02 Reserved. 0x03 4 received channels. 0x04 Reserved. . . . 0xFD 254 received channels. 0xFE Reserved. 0xFF 256 received channels. **Note:** The even values are reserved.
— 15–11	0	Reserved. Write to zero for future compatibility.	
RCDBL 10–8	0	**Receive Channel Data Bits Latency** Defines the maximum amount of receive channel bits stored in the TDM local memory before they are transferred for processing. RCDBL determines the maximum data latency in the following way: Maximum data latency= (RCDBL)/RCS × (receive frame duration). For details, see **Section 19.2.6**. **Notes:** **1.** The maximum data latency is the latency at the worst case when the bus is very loaded. Typically the latency it much smaller. **2.** RCS is field at RFP register defines the channel size. **3.** The minimum number of receive channel is limited if the RCDBL field is clear.The minimum receive number of channels is 128/(receive channel size) + 2.	000 Maximum 64 channel bits. 001 Maximum 128 channel bits. 010 Maximum 256 channel bits. 011 Maximum 512 channel bits. 100 Maximum 1024 channel bits. 101 Maximum 2048 channel bits. 110 Reserved. 111 Reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
RCS 5–2	0	**Receive Channel Size** Determines the receiver channel size – 1. For details, see **Section 19.2.6**.	0000 Reserved. 0001 The receiver channel size is 2 bits. 0010 Reserved. 0011 The receiver channel size is 4 bits. 0100–0110 Reserved. 0111 Receiver channel size is 8 bits. 1000–1110 Reserved. 1111 Receiver channel size is 16 bits.

Table 19-13. TDM*x*RFP Bit Descriptions (Continued)

Name	Reset	Description	Settings
RT1 1	0	**Receive T1 frame** Determines whether the receive frame is T1 frame or non T1. **Note:** In T1 mode the channel size must be 8 bits (RCS = 0x7) and the number of channels must be 24 × (number of links). For example, if the number of link is 2 (RTSAL[1–0] = 01), the number of channels should be 48 (RNCF = 0x2F). For details, see **Section 19.2**.	0 The receive frame is non T1 frame. 1 The receive frame is T1 frame.
RUBM 0	0	**Receive Unified Buffer Mode** Indicates that all the received data is directed to one buffer. When RUBM is set, the number of active links must be 1 (RTSAL = 0b0000 or RTSAL = 0100). The channel parameters of all the receive channels are located in the TDMxRCPR0 (See page 19-61). For details, see **Section 19.2.6.4**. **Note:** When this bit is set, the TDMxRIR[RRDO] bit should be cleared.	0 Each channel is written to a different data buffer in the internal MBus. 1 All the channels are written to the same data buffer in the internal MBus.

Table 19-14 describes the RNCF valid value as a function of the RTSAL[0–1] field.

Table 19-14. RNCF[3–0] Valid Values

RTSAL[1–0]	Number of Active Links	RNCF[3–0] Suffix	Valid Value of RNCF
00	1	xxxxxxx1	The total number of channels must have a granularity of two.
01	2	xxxxxx11	The total number of channels must have a granularity of four.
10	Reserved.		
11	4	xxxxx111	The total number of channels must have a granularity of eight.

19.7.1.5 TDMx Transmit Frame Parameters (TDMxTFP)

TDM*x*TFP TDM*x* Transmit Frame Parameters Offset 0x3FD8

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—								TNCF							
Type	R/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
	—					TCDBL			—		TCS				TT1	TUBM
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TFP defines the TDM*x* transmit frame parameters.

Table 19-15. TDM*x*TFP Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TNCF 23–16	0	**Transmit Number of Channels in a TDM Frame** Specifies the total number of channels that are transmitted in the TDM modules. One TDM frame contains 2–256 channels. **Notes:** **1.** TNCF[8–15] = (number of channels that transmit on one active link) × (number of active data links) – 1. the number of active data links is specified in the RTSAL field. **2.** If TCDBL field is cleared, the minimum number of channels is limit. The minimum transmit number of channels is 128 / (transmit channel size) + 2. for example if the transmit channel size is 16 bits then the transmit TDM frame should contain at least 10 channels. The number of active data links is specified in the RTSAL field. **Table 19-16** describes the TNCF valid value as a function of the TDMxGIR[RTSAL] field (Receive and Transmit Sharing and Active Links). For details, see **Section 19.2.5**.	0x00 Reserved. 0x01 2 Transmit channels. 0x02 Reserved. 0x03 4 Transmit channels. 0x04 Reserved. . . . 0xFE Reserved. 0xFF 256 Transmit channels. The even values are reserved.
— 15–11	0	Reserved. Write to zero for future compatibility.	

Table 19-15. TDM*x*TFP Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCDBL 10–8	0	**Transmit Channel Data Bits Latency** Defines the maximum transmit channel bits that can be stored in the TDM local memory before it transfers output. TCDBL determines the maximum data latency in the following way: Maximum data latency = (TCDBL) / TCS × (transmit frame duration). See **Section 19.2.6**. **Notes:** **1.** The maximum data latency is the latency at the worst case when the bus is very loaded. Typically, actual latency is much smaller. **2.** TDMxTFP[TCS] defines the transmit channel size. **3.** The minimum number of transmit channel is limit if the RCDBL field is clear. The minimum transmit number of channels is 128 / (transmit channel size) + 2.For example see **TNCF** field.	000 Maximum 64 channel bits. 001 Maximum 128 channel bits. 010 Maximum 256 channel bits. 011 Maximum 512 channel bits. 100 Maximum 1024 channel bits. 101 Maximum 2048 channel bits. 110 Reserved. 111 Reserved.
— 7–6	0	Reserved. Write to zero for future compatibility.	
TCS 5–2	0	**Transmit Channel Size** Determines the transmitter channel size – 1. For details, see **Section 19.2.5**.	0000 Reserved 0001 The transmitter channel size is 2 bits. 0010 Reserved 0011 The transmitter channel size is 4 bits. 0100 Reserved. 0101 Reserved. 0110 Reserved. 0111 The transmitter channel size is 8 bits. 1000–1110 Reserved 1111 The transmitter channel size is 16 bits.
TT1 1	0	**Transmit T1 Frame** Determines whether the TDM transmitter drives a T1 frame or non T1 frame. **Note:** In T1 mode, the channel size must be 8 (TCS = 0x7) and the number of channels 24 × (number of links). For example, if the number of links is 1 (RTSAL[1–0] = 00, the number of channels should be 24 (TNCF = 0x17). For details, see **Section 19.2**.	0 Transmit frame is non T1 frame. 1 Transmit frame is T1 frame.
TUBM 0	0	**Transmit Unified Buffer Mode** Indicates that all the transmit data is transferred from one buffer. When TUBM is set, the number of active links must be 1 (RTSAL = 0b0000 or 0b0100). The parameters of all transmit channels are located in TDMxTCPR0. For details, see **Section 19.2.6.4**. **Note:** When this bit is set, the TDMxTIR[TRDO] bit should be cleared.	0 Each channel is read from a different data buffer in the internal MBus. 1 All the channels are read from the same data buffer in the internal MBus.

Table 19-16 describes the TNCF valid value as function of the RTSAL field.

Table 19-16. TNCF[3–0] Valid Values

RTSAL[2–0]	Number of Active Links	TNCF[3–0] Suffix	Valid Value of TNCF
00	1	xxxxxxx1	The total number of channels must have a granularity of two.
01	2	xxxxxx11	The total number of channels must have a granularity of four.
10	Reserved		
11	4	xxxxx111	The total number of channels must have a granularity of eight.

19.7.1.6 TDMx Receive Data Buffer Size (TDMxRDBS)

TDM*x*RDBS TDM*x* Receive Data Buffer Size Offset 0x3FD0

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

TDM*x*RDBS defines the receive data buffers size in bytes.The buffers for A/μ-law channels are double size.

Table 19-17. TDM*x*RDBS Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBS 23–0	0	**Receive Data Buffers Size** Receive data buffers size equals the receive data buffer size in bytes minus 1. The buffer size is aligned to 8 bytes, so bits 0–2 must be set to “111”. For details, see **Section 19.2.6.1**, *Data Buffer Size and A/m-law Channels,* on page 19-23. **Note:** The minimum buffer size is 16 bytes.	0x00000F to 0xFFFFFF

19.7.1.7 TDMx Transmit Data Buffer Size (TDMxTDBS)

TDM*x*TDBS TDM*x* Transmit Data Buffer Size Offset 0x3FC8

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

TDM*x*TDBS defines the transmit data buffer size in bytes. The buffers for A/μ channels are double size.

Table 19-18. TDM*x*TDBS Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBS 23–0	0	**Transmit Data Buffers Size** Transmit data buffers size equals the transmit data buffer size in bytes minus 1. The buffer size is aligned to 8 bytes, so bits 29–31 must be set to “111.” For details, see **Section 19.2.6.1**. **Note:** The minimum buffer size is 16 bytes.	0x00000F to 0xFFFFFF

19.7.1.8 TDMx Receive Global Base Address

TDM*x*RGBA TDM*x* Receive Global Base Address Offset 0x3FC0

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

TDM*x*RGBA determines the 16 most significant bits global base address of the receiver data buffers. The actual address of each receive data buffer is RCDBA + (RGBA << 16). See **Section 19.2.6.2**.

Table 19-19. TDMxRGBA Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.

Table 19-19. TDMxRGBA Bit Descriptions

Name	Reset	Description
RGBA 15–0	0	**Receive Global Base Address** Determines the global base address of the receiver data buffers. It is added to the channel data buffer address and to the current receive displacement to generate the actual data address.

19.7.1.9 TDMx Transmit Global Base Address

TDM*x*TGBA TDM*x* Transmit Global Base Address Offset 0x3FB8

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

TDM*x*TGBA determines the 16 most significant bits global base address of the transmitter data buffers. The actual address of each transmit data buffer is TCDBA + (TGBA << 16). See **Section 19.2.6.2**.

Table 19-20. TDM*x*TGBA Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
TGBA 15–0	0	**Transmit Global Base Address** Determines the global base address of the transmit data buffers. It is added to channel data buffer address and to the current transmit displacement to generate the actual address.

19.7.1.10 TDMx Transmit Force Register (TDMxTFR)

TDM*x*TFR TDM*x* Transmit Force Register 0x3F10

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

TDM*x*TFR sets the priority upgrade value for the transmit channels.

Table 19-21. TDMxTFR Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
PUV 15–0	0	**Priority Upgrade Value** Determines the threshold value for which the TDM transmitter upgrades its system priority. The value is used to avoid an underrun condition. If the value is 0x0000 (reset value), the priority is not upgraded and remains low. The maximum value, which causes the priority always to be high, depends on the value stored in TDMxTFP[TCDBL]. If TDMxTFP[TCDBL] = 0x000, achieve the maximum value is achieved by setting PUV = TDMxTFP[TNCF]. If TDMxTFP[TCDBL] is not 0x000, set the PUV using: PUV = 64/TDMxTFP[TCS] × TDMxTNB[TNB]. See page 19-50 for TDMxTFP details and page 19-68 for TDMxTNB details.

19.7.1.11 TDMx Receive Force Register (TDMxRFR)

TDM*x*RFR TDM*x* Receive Force Register0x3F18

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

TDM*x*RFR sets the priority upgrade value for the transmit channels.

Table 19-22. TDMxRFR Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
PUV 15–0	0	**Priority Upgrade Value** Determines the threshold value for which the TDM receiver upgrades its system priority. The value is used to avoid an overrun condition. If the value is 0x0000 (reset value), the priority is not upgraded and remains low. The maximum value, which causes the priority always to be high, depends on the value stored in TDMxRFPx[RCDBL]. If TDMxRFP[RCDBL] = 0x000, achieve the maximum value by setting PUV = TDMxRFP[RNCF]. If TDMxRFP[RCDBL] is not 0x000, set the PUV using PUV = 64/TDMxRFP[RCS] × RNBx[RNB]. See page 19-47 for TDMxRFP details and page 19-67 for TDMxRNB details.

19.7.1.12 TDMx Parity Control Register (TDMxPCR)

TDMxPCR TDMx Parity Control Register Offset 0x3F00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—													PC	PIE	PIL
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDMxPCR controls the parity check operation.

Table 19-23. TDMxPCR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
PC 2	0	**Parity Check** Enables the parity check mechanism. When the bit is set, you can write to TDMxRCPRn[27–24] or TDMxTCPRn[27–24].	0 Parity check is disabled. 1 Parity check is enabled
PIE 1	0	**Parity Interrupt Enable** Enables/disables the parity interrupt.	0 Parity interrupt is disabled. 1 Parity interrupt is enabled
PIL 0	0	**Parity Interrupt Level** Selects the trigger type for the parity interrupt.	0 Parity interrupt is level-triggered. 1 Parity interrupt is edge-triggered.

19.7.2 Control Registers

The following sections describe the TDM control registers.

19.7.2.1 TDMx Adaptation Control Register

TDMxACR TDMx Adaptation Control Register Offset 0x3FB0

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

TDMxACR controls the activation/deactivation of the TDMx adaptation machine. The propagation of the enable/disable to the adaptation machine is delayed because is clocked by the serial clock.

Table 19-25. TDMxACR Bit Descriptions

Name	Reset	Description	Settings
— 31–2	0	Reserved. Write to zero for future compatibility.	
AME 1	0	**Adaptation Machine Enable** Determines whether the adaptation machine is enabled or disabled.	0 Adaptation machine is disabled. 1 Adaptation machine is enabled
LTS 0	0	**Learn Transmit Sync** Determines whether the adaptation machine learns the transmit sync or the receive sync.	0 Adaptation machine learn the receive sync. 1 Adaptation machine learn the transmit sync.

19.7.2.2 TDMx Receive Control Register

TDM*x*RCR TDM*x* Receive Control Register Offset 0x3FA8

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

Reg	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—															REN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RCR controls the activation/deactivation of the TDM*x* Receiver. Receiver activation is valid only when the RENS bit is clear.

Table 19-26. TDM*x*RCR Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
REN 0	0	**Receive Enable** Determines whether the receive TDM is enabled or disabled. **Note:** Setting this bit is the last step in initializing the receiver.	0 Receiver is disabled. 1 Receiver is enabled.

19.7.2.3 TDMx Transmit Control Register (TDMxTCR)

TDM*x*TCR TDM*x* Transmit Control Register Offset 0x3FA0

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

TDM*x*TCR controls the activation/deactivation of the TDM*x* Transmitter. Transmitter activation is valid only when the TENS bit is clear.

Table 19-27. TDM*x*TCR Bit Descriptions

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
TEN 0	0	**Transmit Enable** Determines whether the transmit TDM is enabled or disabled. Setting this bit is the last step in initializing the transmitter.	0 Transmitter is disabled. 1 Transmitter is enabled.

19.7.2.4 TDMx Receive Data Buffer First Threshold (TDMxRDBFT)

TDM*x*RDBFT TDM*x* Receive Data Buffer First Threshold Offset 0x3F98

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

TDM*x*RDBFT determines the first threshold of the receive data buffers. When the receive buffers are filled up to the first threshold defined by this field—the TDM*x* Receive Data Buffers Displacement Register (TDM*x*RDBDR) = TDM*x* Receive Data Buffers First Threshold (TDM*x*RDBFT) + 8—the RFTE bit in the TDMx Receive Event Register (TDM*x* RER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated at any time, even when the TDM*x* receiver is enabled. For details, see **Section 19.2.6.3**.

Table 19-28. TDM*x*RDBFT Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBFT 23–0	0	**Receive Data Buffer First Threshold** Determines the location of the first threshold in the receive data buffers. The register value has a granularity of 8 bytes; that is, the three LSBits are always clear.	0x000000 to (RDBS – 7)

19.7.2.5 TDMx Transmit Data Buffer First Threshold

TDM*x*TDBFT TDM*x* Transmit Data Buffer First Threshold Offset 0x3F90

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

TDM*x*TDBFT determines the first threshold of the transmit data buffers. When the transmit buffers are emptied up to the first threshold defined by this field—the TDM*x* Transmit Data Buffers Displacement Register (TDM*x*TDBDR) = the TDM*x* Transmit Data Buffers First Threshold (TDM*x*TDBFT) + 8—the TFTE bit in the TDM*x* Transmit Event Register (TDM*x*TER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated any time, even when the TDMx transmitter is enabled. For details, see **Section 19.2.6.3.**

Table 19-29. TDM*x*TDBFT Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBFT 23–0	0	**Transmit Data Buffer First Threshold** Determines the location of the first threshold in the transmit data buffers. The register value has a granularity of eight bytes; that is, the three LSBits are always clear.	0x000000 to (TDBS – 7)

19.7.2.6 TDMx Receive Data Buffer Second Threshold

TDM*x*RDBST TDM*x* Receive Data Buffer Second Threshold Offset 0x3F88

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

TDM*x*RDBST determines the second threshold of the receive data buffers. When the receive buffers are filled up to the second threshold defined by this field—the Receive Data Buffers Displacement Register (TDM*x*RDBDR) = the Receive Data Buffers Second Threshold (TDM*x*RDBST) + 8—the RSTE bit in the TDMx Receive Event Register (TDM*x*RER) is set. If the associated enable bit is also set, an interrupt is generated. This register can be updated any time, even when the TDM*x* receiver is enabled. For details, see **Section 19.2.6.3**.

Table 19-30. TDM*x*RDBFT Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBST 23–0	0	**Receive Data Buffer Second Threshold** Determines the location of the second threshold in the receive data buffers. The register value has a granularity of eight bytes; that is, the three LSBits are always clear.	0x000000 to (RDBS – 7)

19.7.2.7 TDMx Transmit Data Buffer Second Threshold

TDM*x*TDBST TDM*x* Transmit Data Buffer Second Threshold Offset 0x3F80

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TDBST															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TDBST determines the second threshold of the transmit data buffers. When the transmit buffers are emptied up to the second threshold defined by this field—the Transmit Data Buffers Displacement Register (TDM*x*TDBDR) = the Transmit Data Buffers Second Threshold (TDM*x*TDBST) + 8—then the TSTE bit in the TDM*x* Transmit Register (TDM*x*TER) is set. If

the associated enable bit is also set, an interrupt is generated. This register can be updated at any time, even when the TDM*x* transmitter is enabled. For details, see **Section 19.2.6.3**.

Table 19-31. TDM*x*TDBST Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBST 23–0	0	**Transmit Data Buffer Second Threshold** Determines the location of the second threshold in the transmit data buffers. The register value has a granularity of 8 bytes; that is, the three LSBits are always clear.	0x000000 to (TDBS – 7)

19.7.2.8 TDMx Receive Channel Parameter Register n

TDM*x*RCPRn TDM*x* Receive Channel Parameter Register n Offset 0x1000 + n*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RACT	RCONV		—	Reserved				RCDBA							
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

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

Note: The value n is reported in decimal. Convert the value to hexidecimal before multiplying to compute the offset value for a specific register location.

TDM*x*RCPRn determines the parameters for channel 0 to channel 255. The TDM*x*RCPRn[RACT] bit can be changed at any time during the receiver operation. All other fields can only be changed when TDM*x*RCPRn[RACT] is cleared. The read/write access to TDM*x*RCPRn registers can done only to 32 bits, write or read of byte or word is not valid. The register reset value is unknown.

Note: All TDM*x*RCPRn with an index number (n) less than or equal to the TDM*x*RFP[RNCF] bit (see page 19-47) should be valid when setting the corresponding TDM*x*RCR[REN] bit (see page 19-57).

The TDMxRCPRn registers are implemented using a compiled memory, and include support of parity mechanisms that allows detection and correction of one soft error using an interrupt (see TDMxPCR on page 19-56)**.**

Table 19-32. TDM*x*RCPRn Bit Descriptions

Name	Reset	Description	Settings
RACT 31	—	**Receive Channel Active** Set when the receive channel *n* is active.	0 The channel is non-active. 1 The channel is active.

Table 19-32. TDM*x*RCPRn Bit Descriptions (Continued)

Name	Reset	Description	Settings
RCONV 30–29	—	**Receive Channel Convert** Determines the type of the incoming channel n: Transparent, A-law, or µ-Law.	00 Receive channel n is a transparent channel. 01 Receive channel n is a µ-Law channel. 10 Receive channel n is an A-Law channel. 11 Reserved.
— 28	—	Reserved. Write to zero for future compatibility.	
— 27–24	—	These bits are used for parity protection.	
RCDBA 23–0	—	**Receive Channel Data Buffer Base Address** Determines the offset of the data buffer *n* base address from the Receive Global Base Address (RGBA).The RCDBA value should be 16 byte aligned; that is, the four LSB should be 0. For details, see **Section 19.2.6.2**.	0x000000–0xFFFFF0.

19.7.2.9 TDMx Transmit Channel Parameter Register n

TDM*x*TCPRn TDM*x* Transmit Channel Parameter Register n Offset 0x2800 + n*0x4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TACT	TCONV		—	Reserved				TCDBA							
Type	R/W															
Reset	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

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

Note: The value n is reported in decimal. Convert the value to hexidecimal before multiplying to compute the offset value for a specific register location.

The TDM*x*TCPRn registers determine the parameters for channel 0 to channel 255. The registers can change any time during the transmitter operation. The TDM*x*TCPRn[TACT] bit can be changed at any time during the transmitter operation. All other fields can only be changed when TDM*x*TCPRn[TACT] is cleared. The read/write access to TDM*x*TCPRn registers can done only as a 32-bit access. A write or read of a byte or a word is not valid.The register reset value is indeterminate.

Note: All TDM*x*TCPRn with an index number (n) less than or equal to the TDM*x*TFP[TNCF] bit (see page 19-50) should be valid when setting the corresponding TDM*x*TCR[TEN] bit (see page 19-58).

The TDMxTCPRn registers are implemented using a compiled memory, and include support of a parity mechanism that allows detection and correction of one soft error using an interrupt (see TDMxPCR on page 19-56).

Table 19-33. TDM*x*TCPRn Bit Descriptions

Name	Reset	Description	Settings
TACT 31	—	**Transmit Channel Active** Set when the transmit channel n is active.	0 The channel is non-active. 1 The channel is active.
TCONV 30–29	—	**Transmit Channel Convert** Determines the type of the transmit channel n: Transparent, A-law, or µ-Law.	00 Transmit channel *n* is a transparent channel. 01 Transmit channel *n* is a µ-Law channel. 10 Transmit channel *n* is an A-Law channel. 11 Reserved.
— 28	—	Reserved. Write to zero for future compatibility.	
— 27–24	—	These bits are used for parity protection.	
TCDBA 23–0	—	**Transmit Channel Data Buffer Base Address** Determines the offset of the transmit data buffer *n* base address from the Transmit Global Base Address (TGBA). The TCDBA value should be 16 byte aligned; that is, the four LSB should be clear. For details, see **Section 19.2.6.2**.	0x000000–0xFFFFF0.

19.7.2.10 TDMx Receive Interrupt Enable Register (TDMXRIER)

TDM*x*RIER TDM*x* Receive Interrupt Enable Register Offset 0x3F78

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—												RSEEE	OLBEE	RFTEE	RSTEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RIER has the same bit format as the TDM*x*RER registers. If an RIER bit is clear, the corresponding event in the TDM*x*RER registers is masked (see page 19-68).

Table 19-34. TDM*x*RIER Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to zero for future compatibility.	
RSEEE 3	0	**Receive Sync Error Event Enable** Enable assertion of the receive error interrupt when the Receive Sync Error (RSE) bit is set (see page 19-68).	0 Receive sync error is masked. 1 Receive sync error is enabled.

Table 19-34. TDM*x*RIER Bit Descriptions (Continued)

Name	Reset	Description	Settings
OLBEE 2	0	**Overrun Local Buffer Event Enable** Enable assertion of an interrupt when the Overrun Local Buffer Event (OLBE) bit is set (see page 19-68).	0 Overrun Local buffer event is masked. 1 Overrun Local buffer event is enabled.
RFTEE 1	0	**Receive First Threshold Event Enable** Enable assertion of the receive first threshold interrupt when the Receive First threshold Event (RFTE) bit is set (see page 19-68).	0 Receive first threshold interrupt is disabled. 1 Receive first threshold interrupt is enabled.
RSTEE 0	0	**Receive Second Threshold Event Enabled** Enable assertion of the receive second threshold interrupt when the Receive Second Threshold Event (RSTE) bit is set (see page 19-68).	0 Receive second threshold interrupt is disabled. 1 Receive second threshold interrupt is enabled

19.7.2.11 TDMx Transmit Interrupt Enable Register (TDMxTIER)

TDM*x*TIER TDM*x* Transmit Interrupt Enable Register Offset 0x3F70

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—												TSEIE	ULBEE	TFTEE	TSTEE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TIER has the same bit format as the TDM*x*TER registers. If a TDM*x*TIER bit is clear, the corresponding event in the TDM*x*TER is masked (see page 19-69).

Table 19-35. TDM*x*TIER Bit Descriptions

Name	Reset	Description	Settings
— 31–4	0	Reserved. Write to zero for future compatibility.	
TSEIE 3	0	**Transmit Sync Error Event Enabled** Enable assertion of the transmit error interrupt when the Transmit Sync Error (TSE) bit is set. See page 19-69.	0 Transmit sync error interrupt is disabled. 1 Transmit sync error interrupt is enabled.
ULBEE 2	0	**Underrun Local Buffer Event Enabled** Enable assertion of an interrupt when the Underrun Local Buffer Event (ULBE) bit is set. See page 19-69.	0 Underrun Local buffer event is masked. 1 Underrun Local buffer event is enabled.
TFTEE 1	0	**Transmit First Threshold Event Enabled** Enable assertion of the transmit first threshold interrupt when the Transmit First Threshold Event (TSTE) bit is set. See page 19-69.	0 Transmit first threshold interrupt is disabled. 1 Transmit first threshold interrupt is enabled.

Table 19-35. TDM*x*TIER Bit Descriptions

Name	Reset	Description	Settings
TSTEE 0	0	**Transmit Second Threshold Event Enabled** Enable assertion of the transmit second threshold interrupt when the Transmit second Threshold Event (TSTE) bit is set.	0 Transmit second threshold interrupt is disabled. 1 Transmit second threshold interrupt is enabled.

19.7.3 Status Registers

The following sections describe the TDM status registers.

19.7.3.1 TDMx Adaptation Sync Distance Register (TDMxASDR)

TDMxASDR TDMx Adaptation Sync Distance Register Offset 0x3F68

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

TDMxASDR indicates the number of receive/transmit bits between the last two consecutive receive/transmit sync events. The register value updates each time the TDMxASR[AMS] bit is set.

Table 19-36. TDMxASDR Bit Descriptions

Name	Reset	Description	Settings
— 31–11	0	Reserved. Write to zero for future compatibility.	
ASD 10–0	0	Adaptation Sync Distance Indicate the number of bits between the last two consecutive sync events. If the TDMxACR[LTS] bit is set, the ASD field indicates the number of transmit bits between the last two transmit sync events. If the LTS bit is clear, the value indicates the number of receive bits between the last two receive sync events.	0x000 The number of bits between the last two sync events is 1. 0x001 The number of bits between the last two sync events is 2. . . . 0x7FF The number of bits between the last two sync events is 2048.

19.7.3.2 TDMx Receive Data Buffers Displacement Register (TDMxRDBDR)

TDM*x*RDBDR TDM*x* Receive Data Buffers Displacement Register Offset 0x3F60

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

TDM*x*RDBDR points to the current displacement in the receive data buffers where the received data should be written by the TDM DMA. For details, see **Section 19.2.6.2**, *Data Buffer Address*, on page 19-24.

Table 19-37. TDM*x*RDBDR Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
RDBD 23–0	0	**Receive Data Buffer Displacement** Points to the current displacement of the received data in the data buffers. The value is unified to all the transparent channels and is doubled for A/μ law channels.	0 to (RDBS – 7) = Receive Data Buffer Size.

19.7.3.3 TDMx Transmit Data Buffer Displacement Register (TDMxTDBDR)

TDM*x*TDBDR TDM*x* Transmit Data Buffer Displacement Register Offset 0x3F58

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

TDM*x*TDBDR points to the current displacement in the transmit data buffers of the data that should be read by the TDM DMA. For details, see **Section 19.2.6.2**, *Data Buffer Address*, on page 19-24.

Table 19-38. TDMxTDBDR Bit Descriptions

Name	Reset	Description	Settings
— 31–24	0	Reserved. Write to zero for future compatibility.	
TDBD 23–0	0	**Transmit Data Buffer Displacement** Points to the current displacement of the transmit data in the transmit data buffers. The value is unified to all the transparent channels and is doubled for A/μ law channels.	The register value can range from 0x000000 to the Transmit Data Buffer (TDBS – 7).

19.7.3.4 TDMx Receive Number of Buffers (TDMxRNB)

TDM*x*RNB TDM*x* Receive Number of Buffers Offset 0x3F50

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

TDM*x*RNB holds the number of buffers in the TDM receive local buffer. Using this register, you can calculate the location of all the bytes belonging to any channel in the TDM local memory.

Table 19-39. TDMxRNB Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	
RNB 4–0	0	**Receive Number of Buffers** Holds the number of buffers in the TDM receive local buffer. For details, see **Section 19.2.5**, *TDM Local Memory*, on page 19-22. **Notes:** **1.** The number of receive buffers equals RNB + 1. **2.** If TDMxRFP[RUBM] = 1, the RNB value is determined by the value of TDMxRFP[RCDBL].	0x00 1 buffer. 0x01 2 buffers. 0x03 4 buffers. 0x07 8 buffers. 0x0F 16 buffers. 0x1F 32 buffers. The other values are reserved.

19.7.3.5 TDMx Transmitter Number of Buffers (TDMxTNB)

TDM*x*TNB TDM*x* Transmitter Number of Buffers Offset 0x3F48

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

TDMxTNB holds the number of transmit buffers in the TDM transmit local buffer.

Table 19-40. TDMxTNB Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	
TNB 4–0	0	**Transmit Number of Buffers** Holds the number of buffers in the TDM transmit local buffer. **Notes:** **1.** The number of transmit buffers equals TNB + 1. **2.** If TDMxTFP[TUBM] = 1, the TNB value is determined by the value of TDMxTFP[TCDBL].	0x00 1 buffer. 0x01 2 buffers. 0x03 4 buffers. 0x07 8 buffers. 0x0F 16 buffers. 0x1F 32 buffers. The other values are reserved.

19.7.3.6 TDMx Receive Event Register (TDMxRER)

TDM*x*RER TDM*x* Receive Event Register Offset 0x3F40

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—											RFSTI	RSE	OLBE	RFTE	RSTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*RER contains the status of the receive data buffers and general receive events. The register can be read at any time. Bits 3–0 are cleared by writing ones to them; writing zero has no effect.

Table 19-41. TDM*x*RER Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	

Table 19-41. TDM*x*RER Bit Descriptions (Continued)

Name	Reset	Description	Settings
RFSTI 4	0	**Receive First or Second Threshold Interrupt Indication** Indicates whether the last receive threshold interrupt is the first or second threshold.	0 Second threshold interrupt. 1 First threshold interrupt.
RSE 3	0	**Receive Sync Error** Indicates whether a sync error has occurred. RSE is set when the receive frame synchronization is lost (the synchronization state change from SYNC to HUNT state) because that a frame sync arrive early or it not recognized at the expected position. During operation, this bit indicates errors on the receive signals of the TDM module. For details, see **Section 19.2.4.3**	0 Normal operation. No receive error has occurred. 1 Receive sync error has occurred.
OLBE 2	0	**Overrun Local Buffer Event** Indicates whether an overrun event has occurred in TDM local memory. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the internal MBus and therefore cannot write the data into the destination memory (data buffer). For details, see **Section 19.2.6**.	0 No overrun event has occurred in the TDM local memory. 1 An overrun event has occurred in the TDM local memory.
RFTE 1	0	**Receive First Threshold Event** This field is set when the first thresholds of all the received data buffers are filled with received data. The first threshold pointer is determined by the Receive Data Buffer First Threshold field (RDBFT). For details, see **Section 19.2.6.3**.	0 No receive first threshold event has occurred. 1 A receive first threshold event has occurred.
RSTE 0	0	**Receive Second Threshold Event** This field is set when the second thresholds of all the receive data buffers are filled with received data. The second threshold pointer is determined by the Receive Data Buffer Second Threshold. (RDBST) field. For details, see **Section 19.2.6**	0 No receive second threshold event has occurred. 1 A receive second threshold event has occurred.

19.7.3.7 TDMx Transmit Event Register (TDMxTER)

TDM*x*TER TDM*x* Transmit Event Register Offset 0x3F38

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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**
	—											TFSTI	TSE	ULBE	TFTE	TSTE
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*TER contains the status of the transmit data buffers and general transmit events.The register can be read at any time. Bits 3–0 are cleared by writing ones to them; writing zero has no effect.

Table 19-42. TDM*x*TER Bit Descriptions

Name	Reset	Description	Settings
— 31–5	0	Reserved. Write to zero for future compatibility.	
TFSTI 4	0	**Transmit First or Second Threshold Interrupt Indication** Indicates whether the last receive threshold interrupt is the first or second threshold.	0 Second threshold interrupt. 1 First threshold interrupt.
TSE 3	0	**Transmit Sync Error** Indicates whether a sync error has occurred. TSE is set when the transmit frame synchronization is lost (the synchronization state change from SYNC to HUNT state) because that a transmit frame sync arrive early or it not recognized at the expected position. During operation, this bit indicates errors on the transmit signals of the TDM module. For details, see **Section 19.2.4.3**	0 Normal operation. No transmit sync error has occurred. 1 A transmit sync error has occurred.
ULBE 2	0	**Underrun Local Buffer Event** Indicates whether an underrun event has occurred in the TDM local buffer. This error should not occur during normal operation. It indicates that the TDM has not received enough bandwidth on the internal MBus and therefore cannot read the data from the data buffers to the TDM local memory. For details, see **Section 19.2.6**.	0 No underrun event has occurred in the TDM local memory. 1 An underrun event has occurred in the TDM local memory.
TFTE 1	0	**Transmit First Threshold Event** Indicates whether a first threshold event has occurred. TFTE is set when the first threshold of all the transmit data buffers is empty. The first threshold pointer is determined by the Transmit First Threshold Register (TDMxTFTR). For details, see **Section 19.2.6.3**.	0 No transmit first threshold event has occurred. 1 A transmit first threshold event has occurred.
TSTE 0	0	**Transmit Second Threshold Event** Indicates whether a transmit second threshold event has occurred. TSTE is set when the second threshold of all the transmit data buffers is empty. The second threshold pointer is determined by the transmit Second Threshold Register (TDMxTSTR). For details, see **Section 19.2.6.3**.	0 No transmit second threshold event has occurred. 1 A transmit second threshold event has occurred.

19.7.3.8 TDMx Adaptation Status Register (TDMxASR)

TDM*x*ASR TDMx Adaptation Status Register Offset 0x3F30

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

TDMxASR contains the status of the adaptation machine. The register can be read at any time. Bits are cleared by writing ones to them; writing zero has no effect.

Name	Reset	Description	Settings
— 31–1	0	Reserved. Write to zero for future compatibility.	
AMS 0	0	**Adaptation Machine Status** Indicates the status of the adaptation machine. If the bit is set, new sync arrive and the Adaptation Sync Distance Register (TDMxASDR) loads the new value.	0 No sync arrives and TDMxASDR does not contain a new value. 1 New sync arrives and the TDMxASDR register contains a new value that the SC3850 core should read.

19.7.3.9 TDMx Receive Status Register (TDMxRSR)

TDM*x*RSR TDM*x* Receive Status Register Offset 0x3F28

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

TDM*x*RSR contains the receiver statues. It indicates whether the receiver is synchronized on the receive sync and the receiver is enabled or disabled.

Table 19-43. TDM*x*RSR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
RSSS 2–1	0	**Receive Sync Synchronization Status** Indicates the status of the receive sync synchronization. When the synchronization state is SYNC, the serial part synchronized on the received sync and the received data transfer to the buffer in main memory for processing. **Note:** For details, see **Section 19.2.4.3**.	00 HUNT state. 01 WAIT state. 11 PRESYNC state. 10 SYNC state.
RENS 0	0	**Receive Enable Status** Indicates whether all the receiver parts are enabled/disabled. The propagation of the enable/disable may be delayed because of the different clocks domains.	0 The receiver machine is disabled. 1 The receiver machine is enabled.

19.7.3.10 TDMx Transmit Status Register (TDMxTSR)

TDM*x*TSR TDM*x* Transmit Status Register Offset 0x3F20

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

TDM*x*TSR contains the status of the transmitter. It indicates whether the transmitter is synchronized on the transmit sync and whether it is enabled or disabled.

Table 19-44. TDM*x*TSR Bit Descriptions

Name	Reset	Description	Settings
— 31–3	0	Reserved. Write to zero for future compatibility.	
TSSS 2–1	0	**Transmit Sync Synchronization Status** Indicates the transmit sync synchronization status. When the synchronization state is SYNC, the serial part is synchronized on the transmit sync and new transit data is driven out. For details, see **Section 19.2.4.3**.	00 HUNT state. 01 WAIT state. 11 PRESYNC state. 10 SYNC state.
TENS 0	0	**Transmit Enable Status** Indicates whether all the transmitter parts are enabled/disabled. The propagation of the enable/disable may be delayed because of the different clock domains.	0 The transmit machine is disabled. 1 The transmit machine is enabled.

19.7.3.11 TDMx Parity Error Register (TDMxPER)

TDM*x*PER TDM*x* Parity Error Register Offset 0x3F08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—														PME	PERR
Type	R															
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
	—						PEA									
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM*x*PER contains the parity error status information. It indicates whether a parity error has occurred, whether multiple errors have occurred, and the address of the last error. **Table 19-45** lists the bit field definitions.

Table 19-45. TDM*x*PER Bit Descriptions

Name	Reset	Description	Settings
— 31–18	0	Reserved. Write to zero for future compatibility.	
PME 17	0	**Parity Multiple Error** Indicates whether multiple parity errors occurred before clearing the PERR field. Clearing the PERR field clears this bit.	0 < 2 parity errors occurred. 1 2 or more parity errors occurred.
PERR 16	0	**Parity Error** Indicates whether a parity error occurred. The bit is cleared by writing a 1 to it. Writing a zero has no effect. The parity error is calculated in row resolution (2 channel parameters). Thus, even when accessing a channel parameter with no parity error, this bit indicates a parity error if the other channel has a parity error. Moreover, the parity error is indicated even if a channel is non-active.	0 No parity error occurred. 1 Parity error occurred.
— 15–10	0	Reserved. Write to zero for future compatibility.	
PEA 9–0	0	**Parity Error Address** Internal memory address where the last parity error occurred. The field is cleared when the TDMxPER[PERR] field is cleared. Receive parameter addresses fall in the range 0x100–0x17F and transmit parameter addresses fall in the range 0x280–0x2FF. Each address represents two channels. For example, address 0x100 indicates receive channels 0 and 1; address 0x280 indicates transmit channels 0 and 1.	

UART 20

The UART, also known as the serial communication interface (SCI), is a full-duplex port for serial communications with other devices. This interface uses two dedicated signals: transmit data (UART_TXD) and receive data (UART_RXD) (see **Figure 20-1**). The external connections shared by these signals with GPIO[28–29] are available as general-purpose I/O (GPIO) signals when they are not configured for UART operation (see **Chapter 22**, *GPIO* for details).

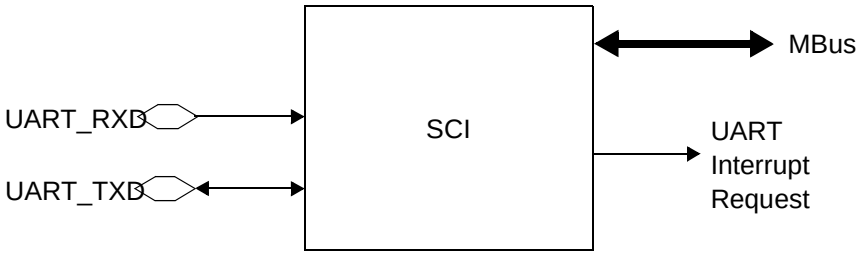


Figure 20-1. UART Interface

The UART is accessible, via the MBus, to an external host or to each of the SC3850 cores. The UART generates one interrupt signal that connects to each SC3850 core so that each SC3850 core can service UART interrupts. For details on UART interrupt signal connectivity, refer to **Chapter 13**, *Interrupt Handling*. When accepting an interrupt request, an SC3850 core or external host should read the UART status register (SCISR) to identify the interrupt source and service it accordingly. During reception, the UART generates an interrupt request when a new character is available to the UART data register, SCI Data Register (SCIDR). An SC3850 core or external host then reads the character from the data register. During transmission, the UART generates an interrupt request when its data register can be written with new character. An SC3850 core or external host then writes the character to the data register.

As **Figure 20-2** shows, the UART allows full duplex, asynchronous, non-return-to-zero (NRZ) serial communication between the MSC8156E and remote devices, including other MSC8156E devices. The UART transmitter and receiver operate independently, although they use the same baud-rate generator and same character length. An SC3850 core monitors the status of the UART, writes the data to be transmitted, and processes received data.

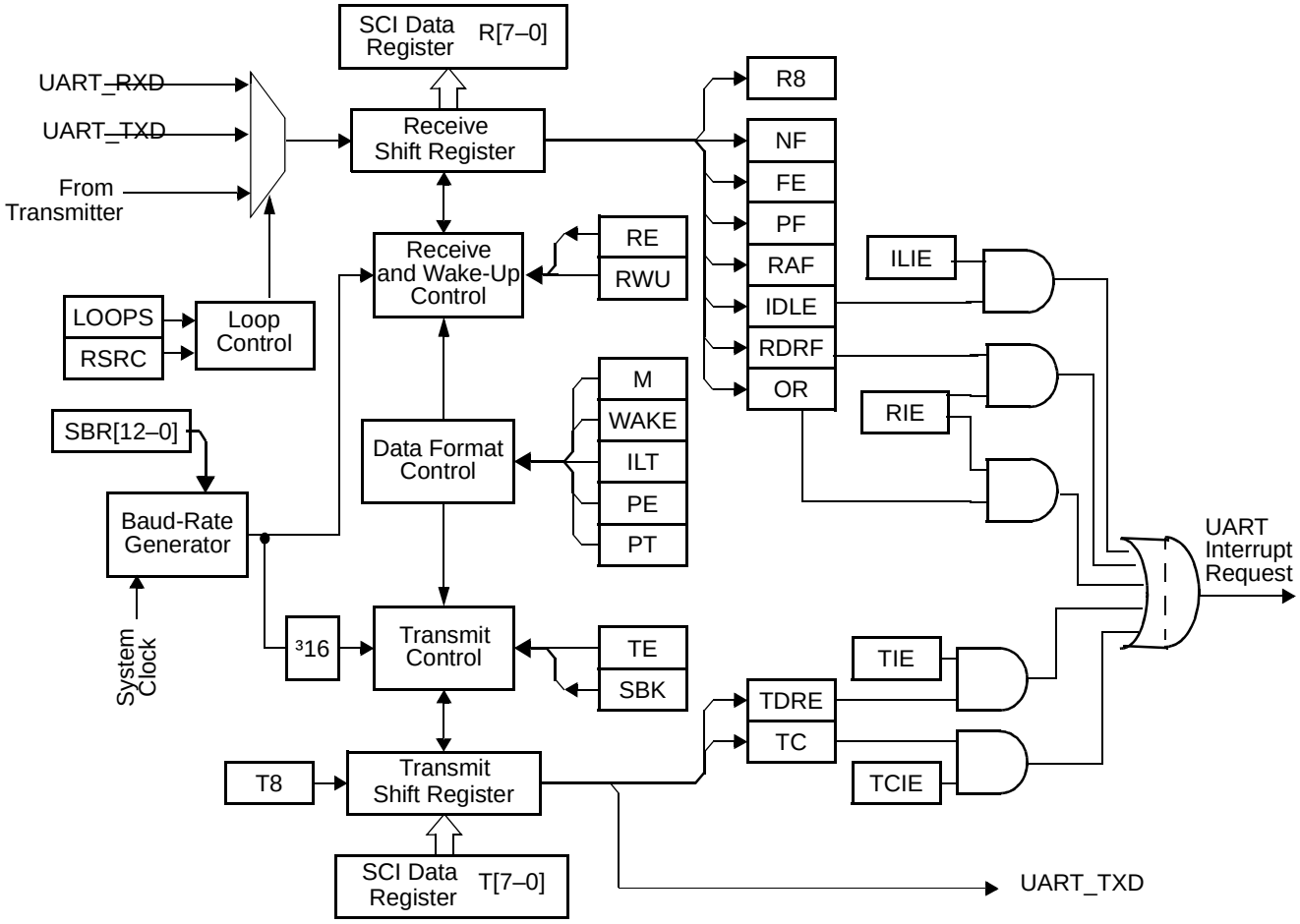
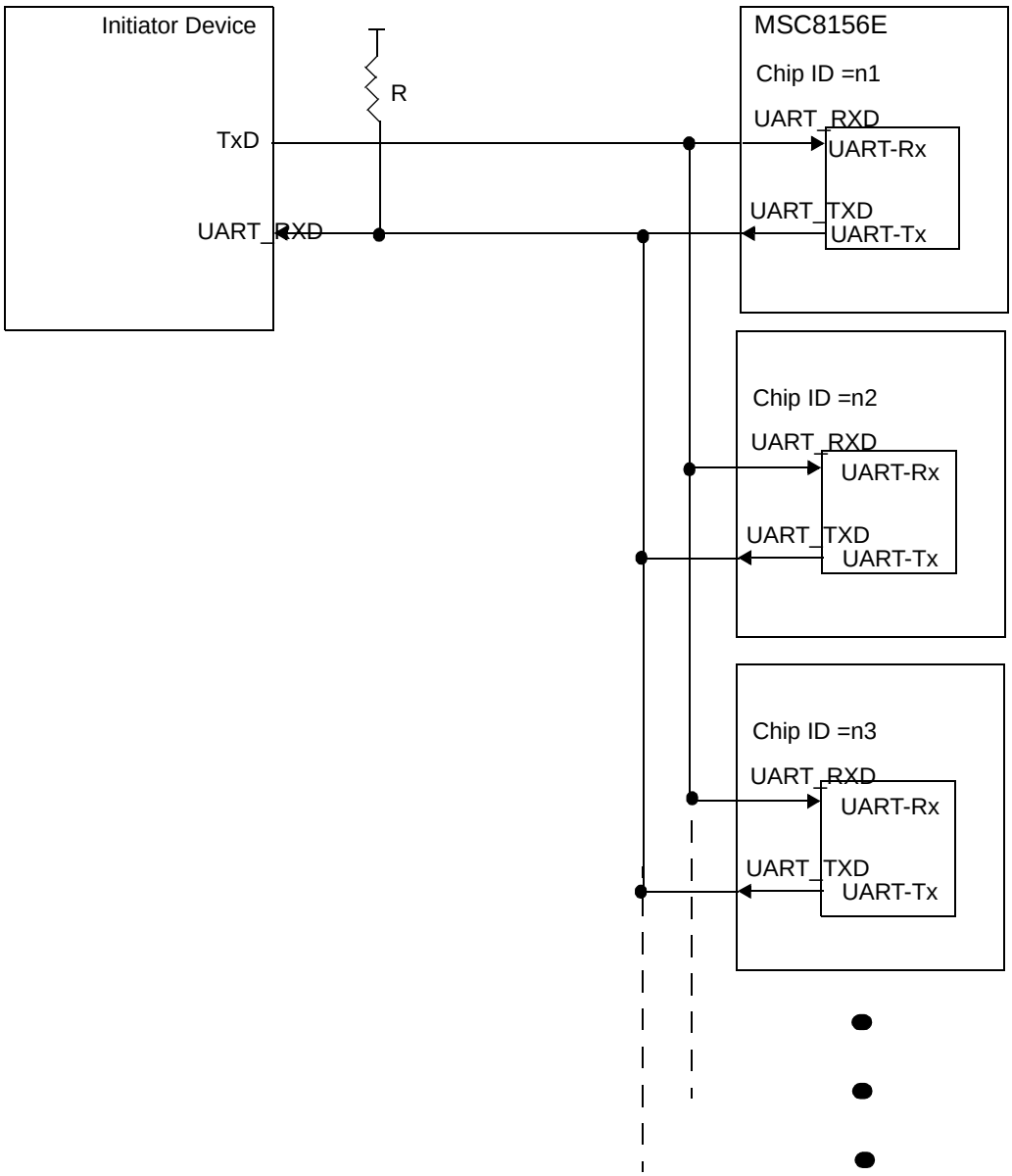


Figure 20-2. UART Block Diagram

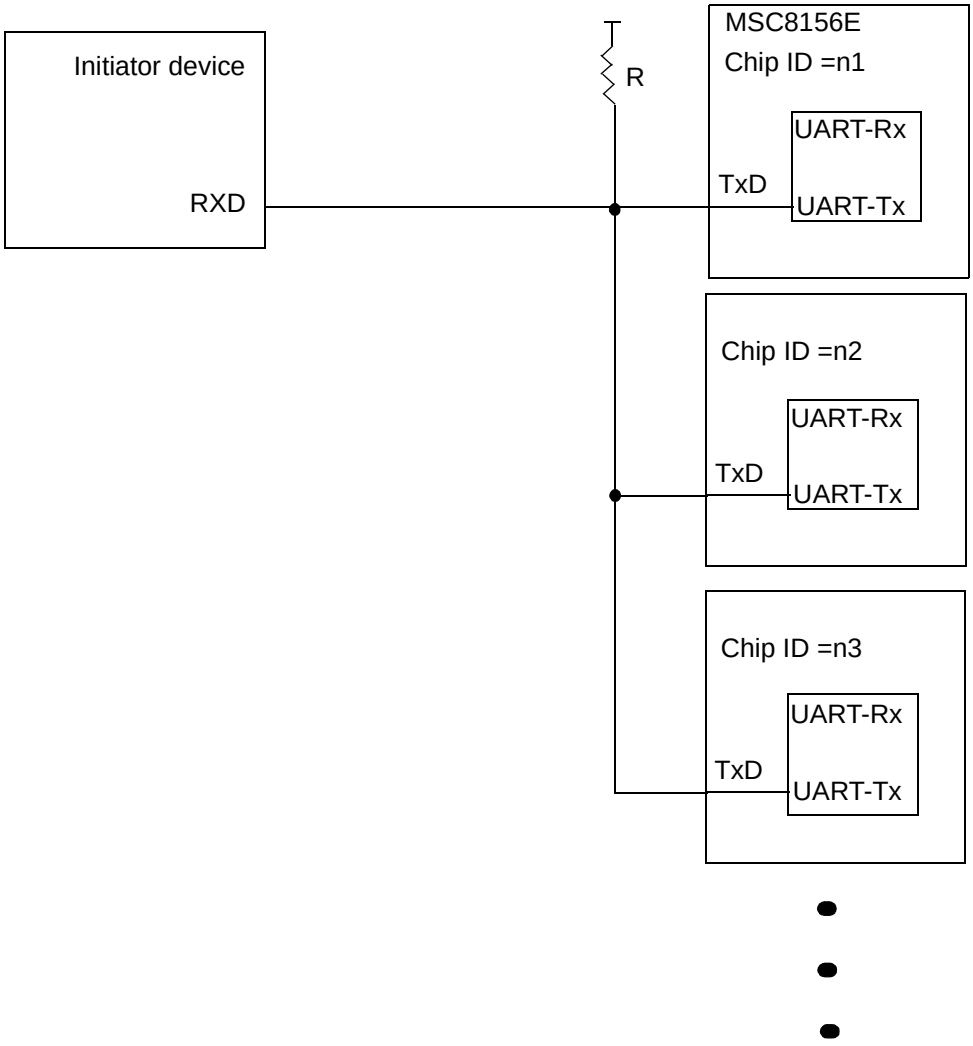
Figure 20-3 shows the full duplex UART system in which the MSC8156E UART transmits and receives simultaneously. A higher-level protocol should handle the full duplex communication to guarantee that no more than one target UART transmits to the UART_RXD signal of the initiator at a given time. Receiver wake-up can obtain such a protocol (see **Section 21.2.7**, *Receiver Wake-Up*). The UART UART_TXD signal can be configured with full CMOS drive or with open-drain drive (see **Chapter 22**, *GPIO*). In both cases, the external pull-up resistor is needed to avoid floating input at the UART_RXD of the initiator.



Note: The RC value on the MultiPoint TxD may limit system baud rate.

Figure 20-3. Full Duplex Multiple UART System

Figure 20-4 shows the UART on a single-wire connection of a half duplex system. The UART_TXD signal must be configured with open-drain drive (see **Chapter 22**, *GPIO*) and an external pull-up resistor. For details on single-wire, see **Section 21.4.2**, *Single-Wire Operation*.



Note: The RC value on the MultiPoint UART_TXD might limit system baud rate.

Figure 20-4. Single-Wire Connection

The UART uses the standard NRZ mark/space data format illustrated in **Figure 20-5**.

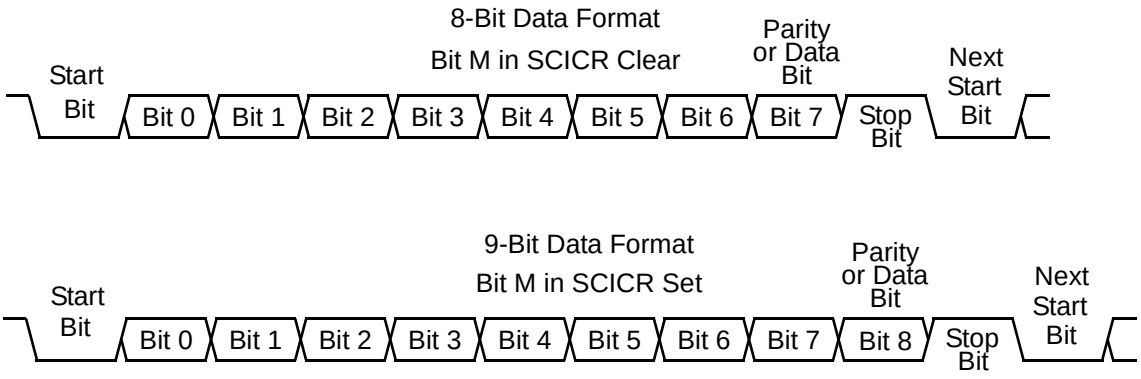


Figure 20-5. UART Data Formats

Each data character is contained in a frame that includes a start bit, eight or nine data bits, and a stop bit. Clearing the M bit in SCI Control Register 1 (SCICR) configures the SCI for 8-bit data characters. A frame with eight data bits has a total of 10 bits, including a start bit and a stop bit.

Table 20-1. Examples of 8-Bit Data Format

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1

Note: The address bit identifies the frame as an address character. The address bit is bit 7 (M = 0) or bit 8 (M = 1) See **Section 21.2.7**, *Receiver Wake-Up*.

Setting the M bit configures the UART for nine-bit data characters. When the UART is configured for 9-bit data characters, the ninth data bit is the T8 or R8 bit in the SCIDR. T8 remains unchanged after transmission and can be used repeatedly without rewriting it. A frame with nine data bits has a total of 11 bits, including a start bit and a stop bit.

Table 20-2. Example of 9-Bit Data Format

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Note: The address bit identifies the frame as an address character. The address bit is bit 7 (M = 0) or bit 8 (M = 1). See **Section 21.2.7**, *Receiver Wake-Up*.

A 13-bit modulus counter in the baud-rate generator derives the baud rate for both the receiver and the transmitter. A value ranging from 1 to 8191 is written to the SBR[12–0] bits to determine the CLASS clock/2 clock divisor. Writing a 0 to SBR[12–0] disables the baud-rate generator. The SBR bits are in the SCI Baud-Rate Register (SCIBR). The baud-rate clock is synchronized with the bus clock and drives the receiver. The baud-rate clock divided by 16 drives the transmitter. The receiver has an acquisition rate of 16 samples per bit time. Baud-rate generation is subject to two sources of error:

- Integer division of the CLASS clock/2 may not give the exact target frequency.
- Synchronization with the bus clock can cause phase shift.

Refer to **Section 21.2.2**, *Data Sampling*, for details on adjusting to the received baud rate at the receiver.

Table 20-3 lists some examples of achieving target baud rates with a CLASS clock/2 frequency of 250 MHz, using the following formula:

$$UART\ baud\ rate = System\ clock / (16 \times SCIBR[12–0])$$

Table 20-3. Baud Rates (CLASS clock/2 = 250 MHz)

Bits SBR	Receiver Clock (Hz)	Transmitter Clock (Hz)	Target Baud Rate	Error (Percentage)
34	7,352,941	459,559	460,000	0.096
68	3,676,471	229,779	230,000	0.096
136	1,838,235	114,890	115,000	0.096
244	1,024,590	64,037	64,000	0.058
407	614,251	38,391	38,400	0.024
814	307,125	19,195	19,200	0.024
1628	153,563	9598	9600	0.024
3255	76,805	4800	4800	0.006
6510	38,402	2400	2400	0.006
Not supported			1200	—

Note: The error relates only to the first source error. Although typically, the CLASS frequency is 500 MHz, it may be reconfigured as indicated in the Clock Modes **Table 7-1** in the Clocks chapter. The divider values must be recomputed for a different CLASS frequency.

20.1 Transmitter

The UART transmitter accommodates either 8-bit or 9-bit data characters. The state of the M bit in the SCI Control Register (SCICR) determines the length of the data characters. When 9-bit data is transmitted, bit T8 in the SCIDR is the ninth bit (bit 8).

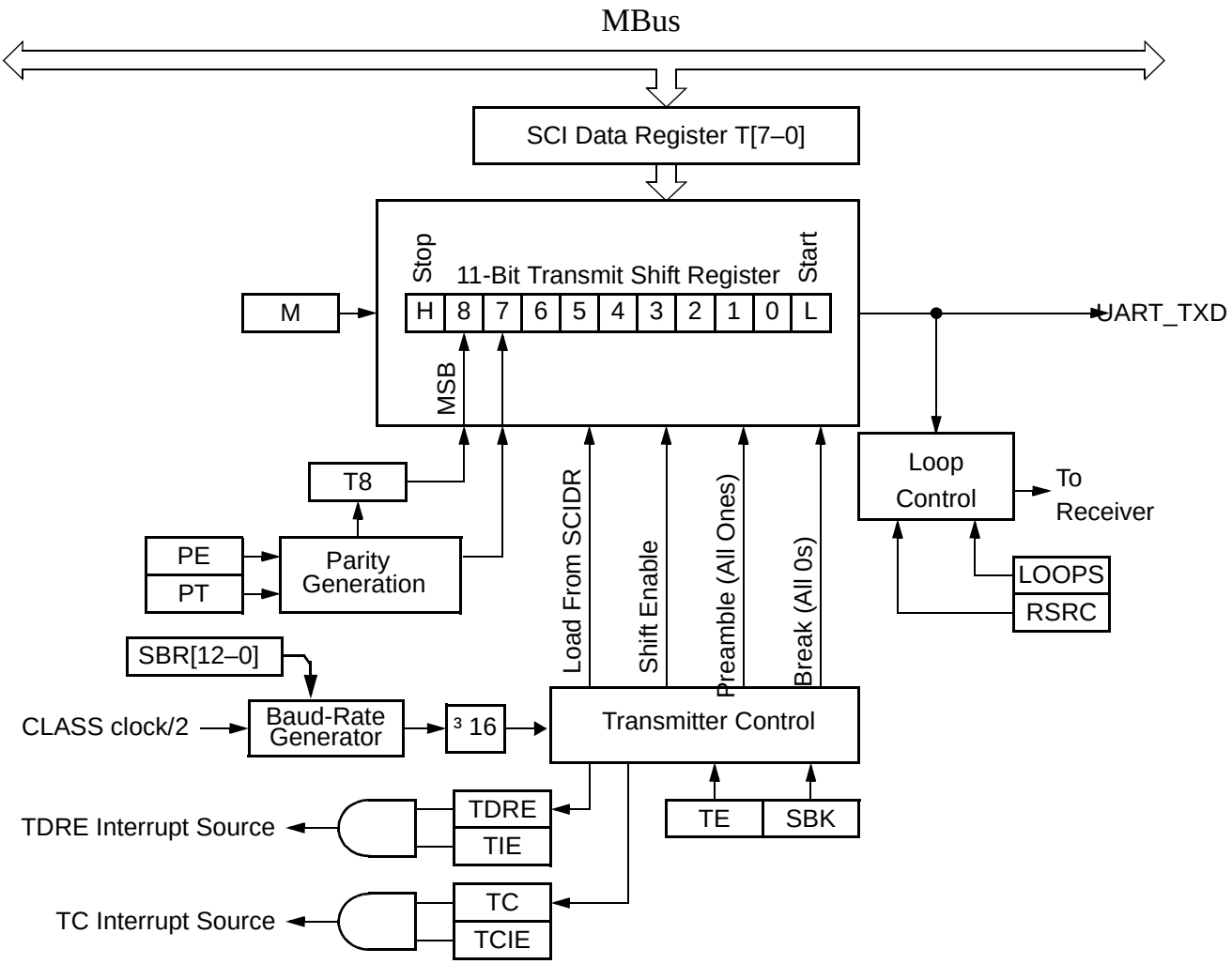


Figure 20-6. Transmitter Block Diagram

20.1.1 Character Transmission

To transmit data, one of the SC3850 cores or an external host writes the data character to the SCI Data Register (SCIDR), which is then transferred to the transmitter shift register. The transmitter shift register then shifts out the data bits on the UART_TXD signal, after it prefaces them with a start bit and appends them with a stop bit. The SCI data register is the write-only buffer between the MBus and the transmit shift register.

The UART also sets a flag, the transmit data register empty flag (TDRE), every time it transfers data from the buffer (SCIDR) to the transmitter shift register. If the Transmit Interrupt Enable (TIE) bit in the SCICR is set, the TDRE flag asserts a UART interrupt request. The transmit interrupt service routine responds to this flag by writing another character to the transmitter buffer (SCIDR), while the shift register is still shifting out the first character. If the TDRE flag is set and no new data or break character transferred to the shift register, the UART sets a flag, transmit complete (TC) and UART_TXD becomes idle.

Begin a UART transmission as follows:

1. Configure the UART:
   a. Select a baud rate. Write the appropriate value to the SCIBR to start the baud-rate generator. Note that the baud-rate generator is disabled when the baud rate is zero. Writing to the 5 MSB (SBR[12–8]) bits of the SCIBR has no effect without also writing to the 8 LSB of SCIBR (SBR[7–0]).
   b. Configure GPIO29 for UART UART_TXD (see **Chapter 22**, *GPIO*):
      - Select the UART transmit signal for the GPIO29 external connection via the GPIO29 configuration registers.
      - Set the direction bit for the GPIO29 port to select output.
   c. Write to the SCICR to configure data length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, PT) and enable the transmit and receive interrupts as required (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK). A preamble character is now shifted out of the transmitter shift register.
2. Perform the transmit procedure for each character:
   d. Poll the TDRE flag by reading the SCISR or responding to the UART interrupt. Keep in mind that the TDRE reset value is one.
   e. If the TDRE flag is set, write the data to be transmitted to SCIDR, where the ninth bit is written to the T8 bit in SCIDR if the UART is in 9-bit data format. Reading TDRE bit in the SCISR and then writing new data to T[7–0] in the SCIDR clears the TDRE flag. Otherwise, the last data transmitted and then UART_TXD goes to idle condition, that is, a logic 1 (high).
3. Repeat step 2 for each subsequent transmission.

Note: The TDRE flag is set when the shift register is loaded with the next data to be transmitted from SCIDR, which occurs 9/16ths of a bit time *after* the start of the stop bit of the previous frame.

Note: When the shift register is empty (the TC and TDRE flags are set), transmission starts no more than one bit time after the data register is written. If only the TC interrupt source is enabled (SCICR[TCIE] = 1, SCICR[TIE] = 0), then you must ensure at least one bit time interval between successive writes to the SCIDR to enable the transmitter software to write twice to the SCIDR per interrupt.

Setting the Transmitter Enable (SCICR[TE]) bit to 1 automatically loads the transmit shift register with a preamble of 10 logic 1s (if M = 0) or 11 logic 1s (if M = 1). After the preamble shifts out, control logic transfers the data from the SCIDR into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

Hardware supports odd or even parity. When parity is enabled, the most significant bit (msb) of the data character is the parity bit.

When the transmit shift register is not transmitting a character, UART_TXD goes to the idle condition, logic 1. When software clears the SCICR[TE] bit, the transmitter relinquishes control of UART_TXD.

Note: If SCIDDR[DDRTX] is set, UART_TXD is driven by a logic 0 (pulled down). Otherwise, if SCIDDR[DDRTX] is cleared, the UART_TXD signal is not driven. See **Chapter 22**, *GPIO* for details about configuring this signal.

If software clears SCICR[TE] while a transmission is in progress (TC = 0), the frame in the transmit shift register continues to shift out. Then the transmitter relinquishes control of UART_TXD even if there is data pending in the SCI data register. To avoid accidentally cutting off the last frame in a message, always wait for TDRE to go high after the last frame before clearing SCICR[TE].

To separate messages with preambles with minimum idle line time, use the following sequence between messages (see also **Figure 20-7**, *Queuing an Idle Character*):

1. Write the last character of the first message to the SCIDR.
2. Wait for the TDRE flag to go high, indicating the transfer of the last frame to the transmit shift register.
3. Insert a preamble by clearing and then setting the SCICR[TE] bit.
4. Write the first character of the second message to the SCIDR.

Another way to separate messages with idle line is to wait until the TC flag is set after writing the last character of the first message to SCIDR, indicating this character has already been transmitted. When TC is set, UART_TXD goes idle. Then, after some idle line time, write the first character of the second message to SCIDR.

20.1.2 Break Characters

Setting the send break bit (SCICR[SBK]) to a value of 1 loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character is ten logic 0s (if M = 0) or eleven logic 0s (if M = 1). As long as SCICR[SBK] is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next frame.

20.1.3 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. The length of idle characters depends on the M bit in SCICR. The preamble is a synchronizing idle character that begins the first transmission initiated after the SCICR[TE] bit is written from 0 to 1. Clearing and then setting the SCICR[TE] bit during a transmission queues an idle character to be sent after the frame currently being transmitted.

Note: When queuing an idle character, return the SCICR[TE] bit to logic 1 before the stop bit of the current frame shifts out to UART_TXD. Setting SCICR[TE] after the stop bit appears on UART_TXD discards data previously written to the SCI data register. Toggle the SCICR[TE] bit for a queued idle character while the TDRE flag is set and immediately before writing the next character to the SCI data register. See **Figure 20-7**, *Queuing an Idle Character*.

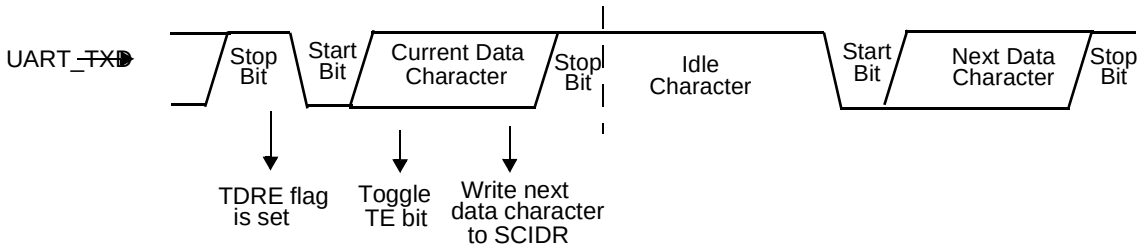


Figure 20-7. Queuing an Idle Character

20.1.4 Parity Bit Generation

The UART can be configured to enable parity bit generation by the parity enable bit (SCICR[PE]). The parity type bit (SCICR[PT]) determines whether to place even or odd parity at T8 (if M = 1) or at T7 (if M = 0) bits of SCIDR.

20.2 Receiver

The SCI receiver can accommodate either 8-bit or 9-bit data characters. The state of the SCICR[M] bit determines the length of data characters. When receiving 9-bit data, bit R8 in the SCIDR is the ninth bit (bit 8).

20.2.1 Character Reception

During a UART reception, the receive shift register shifts a frame in through UART_RXD. The SCI data register is the read-only buffer between the MBus and the receive shift register. After a complete frame shifts into the receive shift register, the data portion of the frame (the character) is transferred to the SCI data register. The receive data register full flag, RDRF, in SCISR is set, indicating that the received character can be read.

The overrun flag, OR, is set when software fails to read the SCIDR before the receive shift register receives the next character. If the receive interrupt enable bit (SCICR[RIE]) is also set, the Receive Data Register Full (RDRF) or the OR flags generate an interrupt request.

Begin an SCI reception as follows:

1. Configure the SCI:
   f. Select the target baud rate and write the appropriate value to the SCI Baud Rate Register (SCIBR). Note that the baud-rate generator is disabled when the baud rate is zero. Writing to 5 MSB bits of SCIBR (SBR[12–8]) has no effect without also writing to 7 LSB of SCIBR (SBR[7–0]).
   g. Configure GPIO28 for UART UART_RXD (see **Chapter 22**, *GPIO*):
   — Select the UART UART_RXD signal as the external connection via the GPIO port 20 configuration registers.
   — Clear the direction bit for GPIO28 in the direction register to select input.
   h. Write to the SCICR to configure data length, parity, and other configuration bits (LOOPS, RSRC, M, WAKE, ILT, PE, PT) and enable the transmitter, interrupts, receive, and wake up as required (TIE, TCIE, RIE, ILIE, TE, RE, RWU, and SBK).If the SBK bit is set, the receiver wakes up if there are particular conditions on the UART_RXD signal according to the WAKE control bit. Refer to **Section 21.2.7**, *Receiver Wake-Up*.
2. Perform the reception procedure for each character:
   i. Poll the RDRF flag by reading the SCISR or responding to the UART interrupt.
   j. If the RDRF flag is set, read the data to be received from SCIDR, where the ninth bit is read from R8 bit in SCIDR if the SCI is in 9-bit data format. Reading RDRF bit at SCISR and then reading new data from SCIDR clears RDRF flag.
3. Repeat step 2 for each subsequent reception.

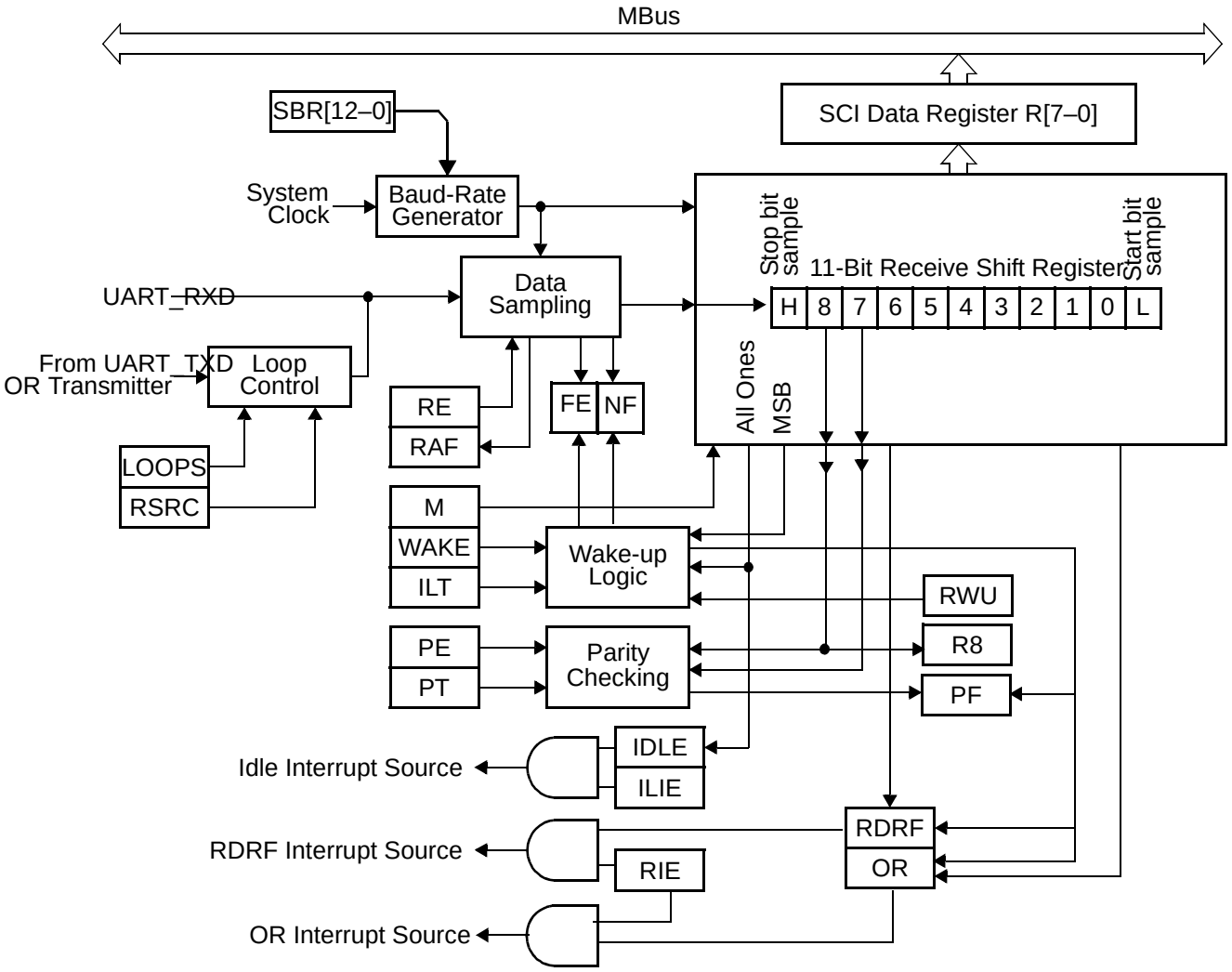


Figure 20-8. UART Receiver Block Diagram

20.2.2 Data Sampling

The receiver samples UART_RXD at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch between the baud rate generated by RT clock and the target baud rate, the RT clock (see **Figure 20-9**) is re-synchronized:

- After every start bit.
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0).

To locate the start bit, data sampling logic searches for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock logic begins to count to 16.

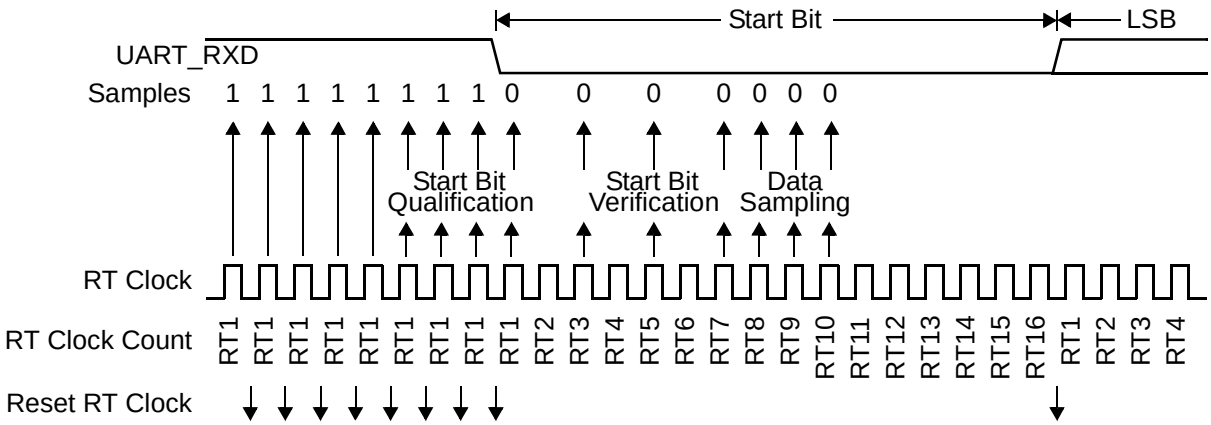


Figure 20-9. Receiver Data Sampling

To verify the start bit and to detect noise, data sampling logic takes samples at RT3 and RT5. If both samples are logic 1 the RT counter is reset and a new search for a start bit begins, else also RT7 sample is taken. If at least two samples (from RT3, RT5, and RT7) are logic 0 then the start bit is perceived. The noise flag, NF, is set if two samples are logic 0 and one is logic 1. **Table 20-4** summarizes the results of the start bit verification samples.

Table 20-4. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
11 (RT7 sample is not taken)	No	0

If start bit verification is not successful, the RT counter is reset and a new search for a start bit begins. To determine the value of a data bit and to detect noise, sample logic takes samples at RT8, RT9, and RT10. The data bit value is determined by the majority of the samples. The noise flag, NF, is set if not all samples have the same logical value. **Table 20-5** summarizes the results of the data bit samples.

Table 20-5. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

Note: The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, sample logic takes samples at RT8, RT9, and RT10. The noise flag, NF, is set if not all samples have the same logical value (the same as for data bit sampling). If the majority of the samples are logic 0 framing error flag, FE, is set. **Table 20-6** summarizes the results of the stop bit samples.

Table 20-6. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In **Figure 20-10** the start bit verification samples RT3 and RT5 determine that the first logic 0 detected is noise and not the beginning of a start bit. The RT counter is reset and the start bit search resumes. The noise flag is not set because the noise occurred before the start bit was found.

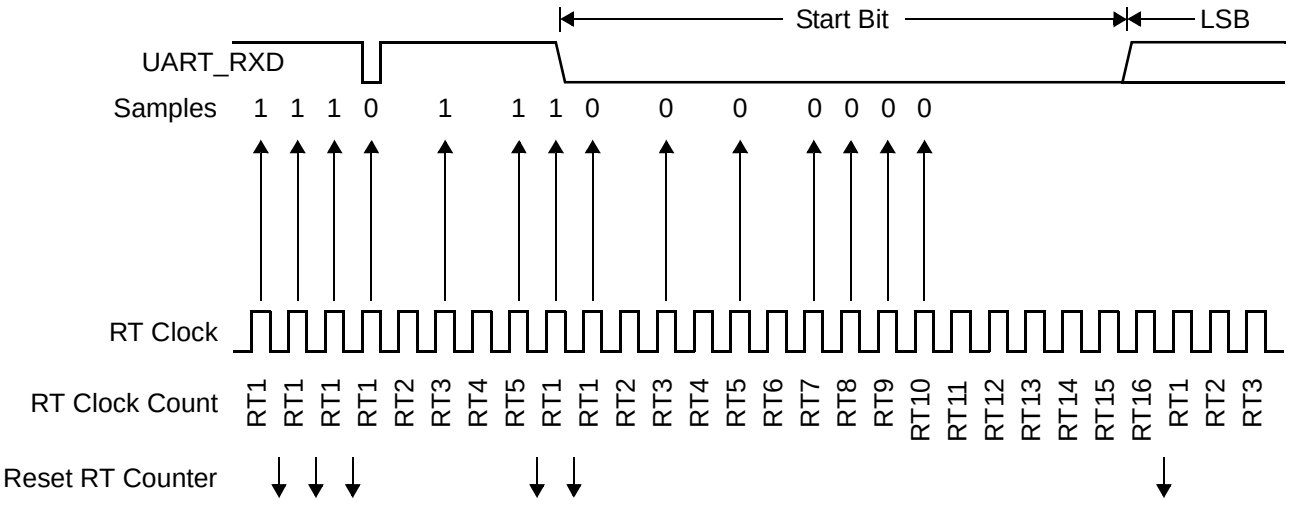


Figure 20-10. Start Bit Search Example 1

In **Figure 20-11**, the verification sample at RT3 is high. The RT3 sample sets the noise flag. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 of the next bit are within the bit time and data recovery is successful.

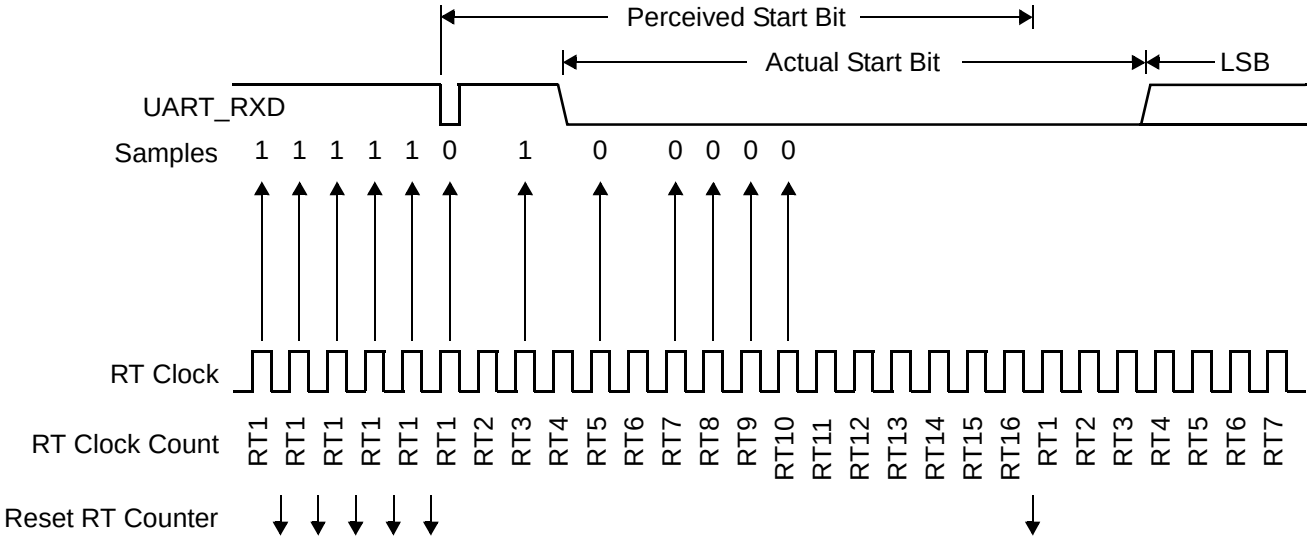


Figure 20-11. Start Bit Search Example 2

In **Figure 20-12** the first start bit verification is a case similar to that in **Figure 20-10**, *Start Bit Search Example 1*, but this time the first logic 0 detected is the real beginning of start bit. The noise is at samples R3 and R5 which causes the RT counter to reset and the start bit search begins again. Although the perceived bit time is misaligned, the data samples RT8, RT9, and RT10 of the next bit are within the bit time and data recovery is successful. For this case the noise and framing error flags are not set.

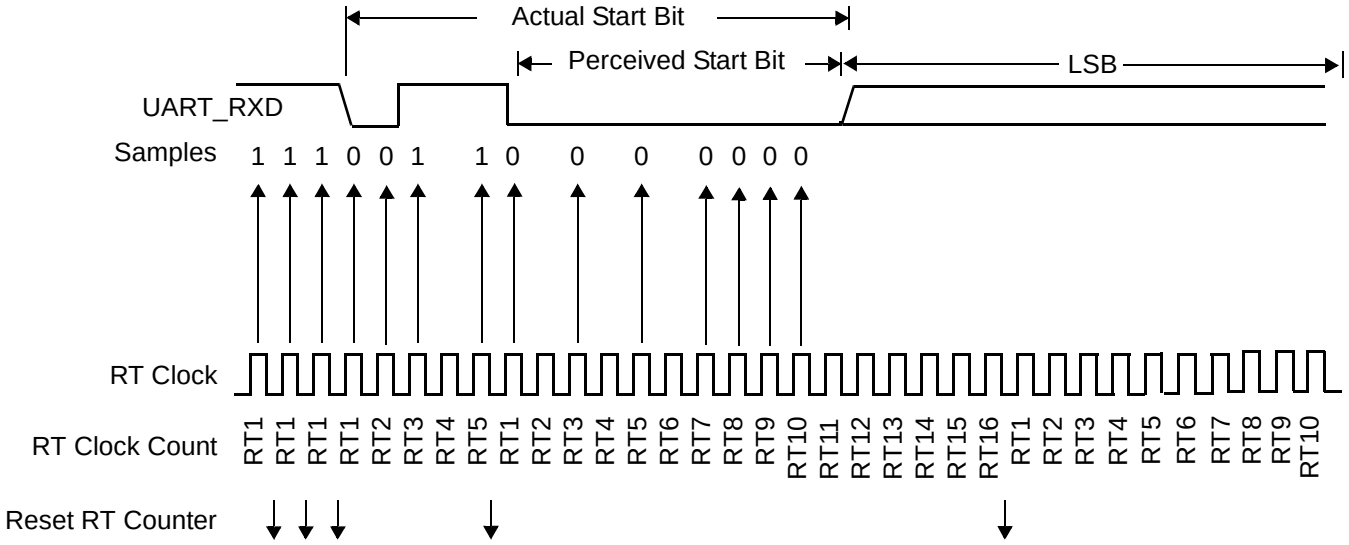


Figure 20-12. Start Bit Search Example 3

In **Figure 20-13**, a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the RT8, RT9, and RT10 data samples of the next bit are within the bit time, and data recovery is successful.

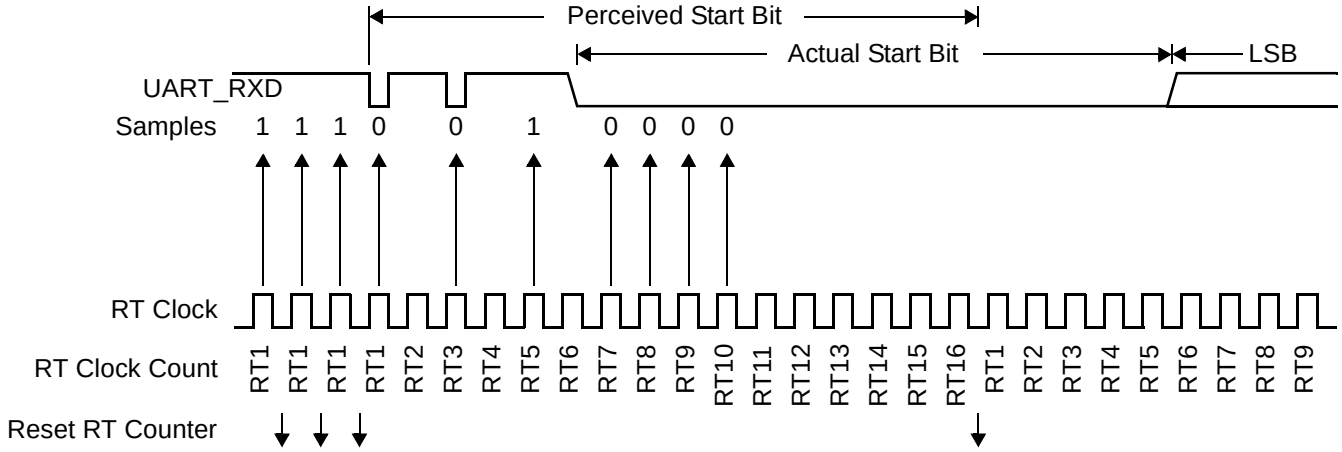


Figure 20-13. Start Bit Search Example 4

Figure 20-14 shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

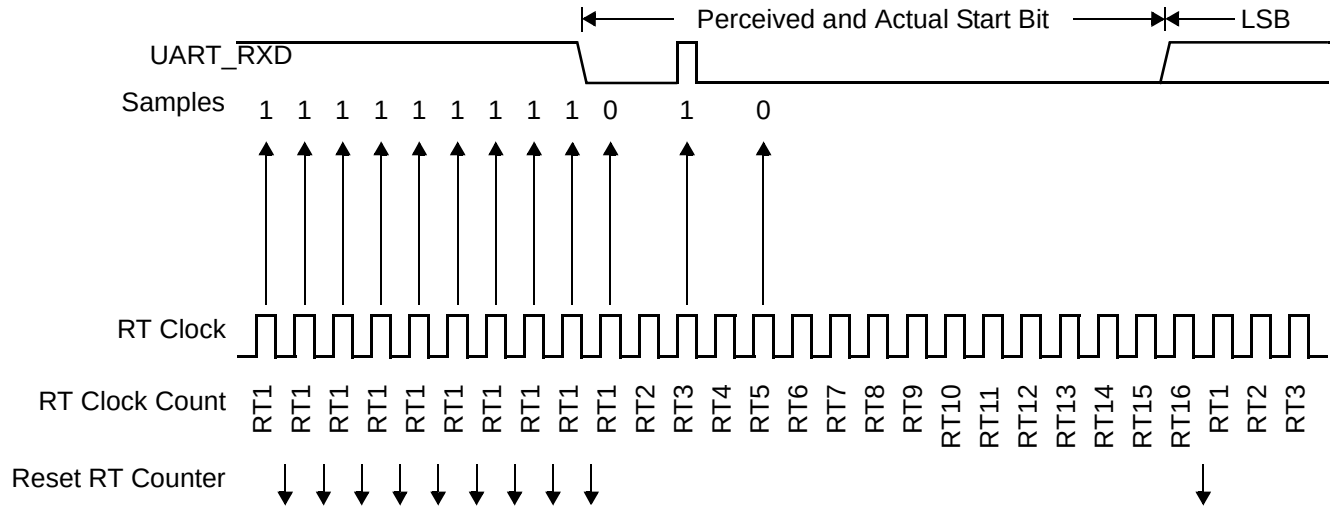


Figure 20-14. Start Bit Search Example 5

Figure 20-15 shows a burst of noise near the beginning of the start bit that causes the start bit not to be found and resets the RT counter. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.

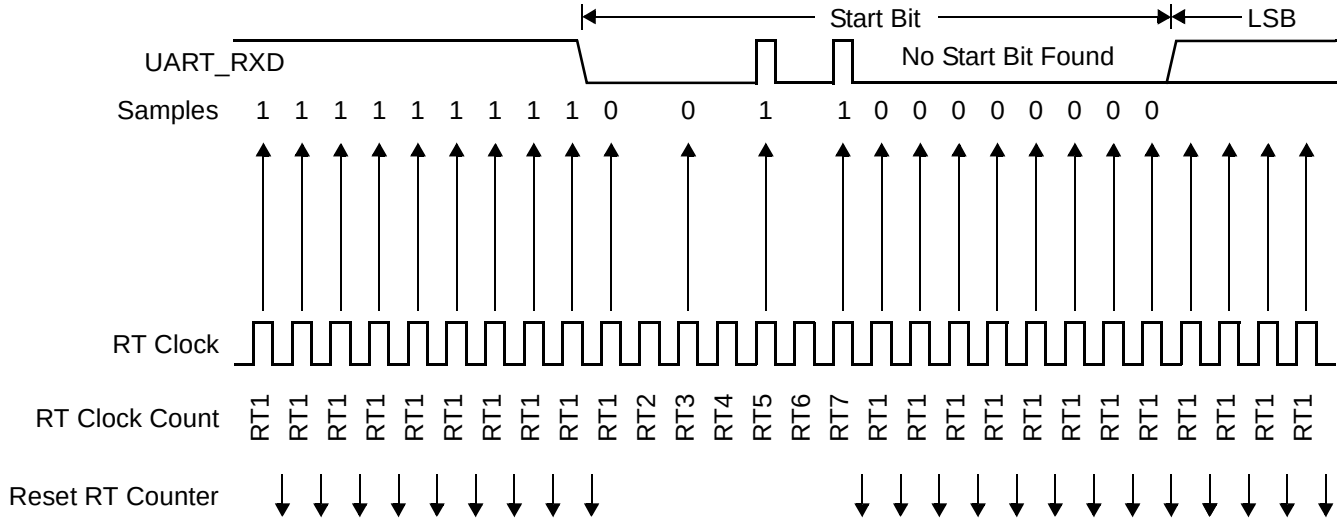


Figure 20-15. Start Bit Search Example 6

In **Figure 20-16**, a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT counter. In the start bits only, the RT8, RT9, and RT10 data samples are not used for determining bit value.

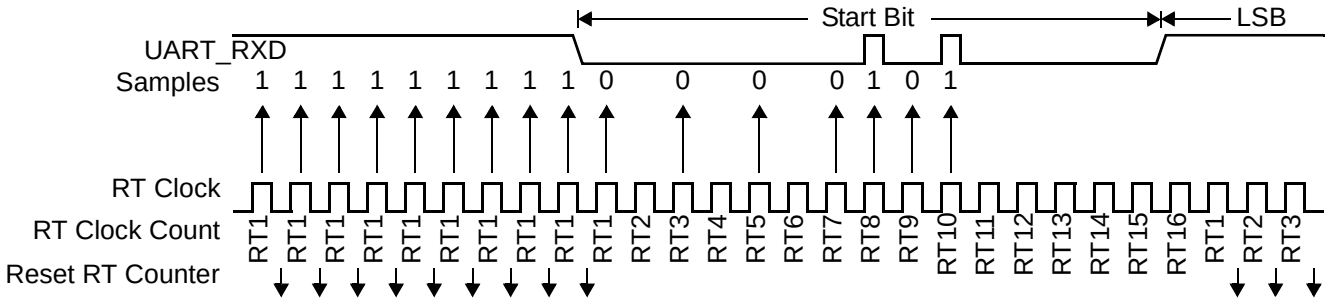


Figure 20-16. Start Bit Search Example 7

20.2.3 Framing Error

If the data sampling logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, SCISR[FE]. A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set. FE inhibits further data reception until it is cleared. Clear SCISR[FE] by reading SCISR and then reading the SCIDR.

20.2.4 Parity Error

The UART can be configured to enable parity check via the Parity Enable (PE) bit in the SCICR. The parity type (SCICR[PT]) determines whether to check for even or odd parity. The Parity Error Flag, SCISR[PF], is set when the parity enable bit is set and the parity of the received character does not match the PT bit. Clear SCISR[PF] by reading the SCISR and then reading SCIDR.

20.2.5 Break Characters

The UART recognizes a break character as a start bit followed by 8 or 9logic 0 data bits and a logic 0 stop bit. Receiving a break character has the following effects on UART registers:

1. The framing error flag (SCISR[FE]) is set.
2. The receive data register full flag (SCISR[RDRF]) is set.

Note: Once the RDRF flag is cleared after being set by a break character, a valid frame must set the RDRF flag again before another break character can set it again.

3. The SCIDR is cleared.
4. The overrun flag (OR), noise flag (NF), parity error flag (PF), or the receiver active flag (RAF) is set (see the discussion in **Section 20.6**).

20.2.6 Baud-Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples (RT8, RT9, and RT10) to fall outside the actual stop bit. A noise error occurs if the RT8, RT9, and RT10 samples are not all the same logical values. A framing error occurs if the receiver clock is misaligned so that the majority of the RT8, RT9, and RT10 stop bit samples are a logic zero. In most applications, the baud-rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming frame, it resynchronizes the RT clock on any valid falling edge within the frame. Resynchronization within frames corrects a misalignment between transmitter bit times and receiver bit times.

20.2.6.1 Slow Data Tolerance

Figure 20-17 shows how much a slow received frame can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

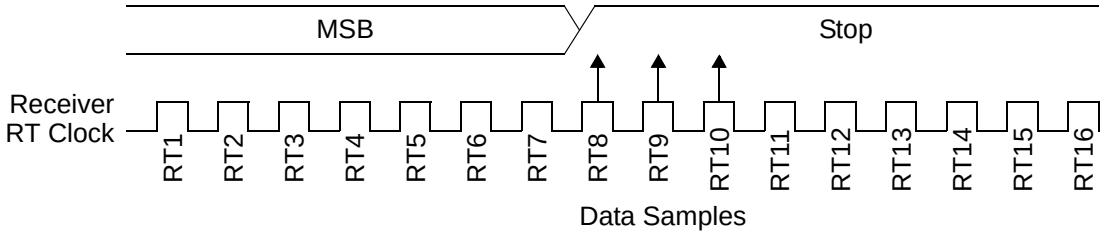


Figure 20-17. Slow Data

For an 8-bit data character, data sampling of the stop bit takes the receiver 9 bit × 16 RT cycles +10 RT cycles =154 RT cycles. With the misaligned character shown in **Figure 20-17**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit × 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit data character with no errors is:

$$((154 - 147) / 154) \times 100 = 4.54\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles. With the misaligned character, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit × 16 RT cycles + 3 RT cycles = 163 RT cycles. The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$((170 - 163) / 170) \times 100 = 4.12\%$$

20.2.6.2 Fast Data Tolerance

Figure 20-18 shows how much a fast received frame can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16, but it is still sampled at RT8, RT9, and RT10.

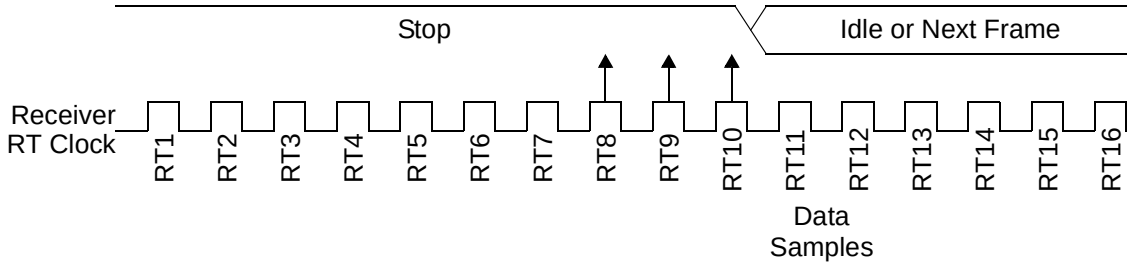


Figure 20-18. Fast Data

For an 8-bit data character, data sampling of the stop bit takes the receiver 9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles. With the misaligned character shown in Figure 20-18, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

$$((154 - 160) / 154) \times 100 = 3.90\%$$

For a 9-bit data character, data sampling of the stop bit takes the receiver 10 bit × 16 RT cycles + 10 RT cycles = 170 RT cycles. With the misaligned character, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit × 16 RT cycles = 176 RT cycles. The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$((170 - 176) / 170) \times 100 = 3.53\%$$

20.2.7 Receiver Wake-Up

The receiver can be put into a standby state, so that the UART (SCI) can ignore transmissions intended only for other receivers in multiple-receiver systems. This is sometimes called putting the receiver to sleep. Setting the receiver wake-up (RWU) bit in the SCICR puts the receiver into a standby state during which receiver interrupts are disabled. The SCI still loads the receive data into the SCIDR, but it does not set the SCISR[RDRF] flag or any other flag. The transmitting device can address messages to selected receivers by including addressing information in the initial frame or frames of each message. Once the receiver is asleep, there must be a wake-up procedure to allow it to respond to messages addressed to it. The SCICR[WAKE] bit determines how the SCI is brought out of the standby state to process an incoming message. This wake bit enables either idle line wake-up or address mark wake-up.

20.2.7.1 Idle Input Line Wake-Up (WAKE = 0)

In idle input line wake-up, an idle condition on UART_RXD (all logic 1s) clears the SCICR[RWU] bit and wakes up the SCI. The initial frame or frames of every message contain addressing information. All receiver software evaluate the addressing information, and receivers for which the message is addressed process the frames that follow. Any receiver for which a message is not addressed can set its SCICR[RWU] bit and return to the standby state. The RWU bit remains set and the receiver remains on standby until another idle character appears on UART_RXD.

Idle line wake-up requires that messages be separated by at least one idle character and that no message contain idle characters. The idle character that wakes a receiver does not set the receiver idle bit, IDLE, or the receive data register full flag, SCISR[RDRF]. The idle line type bit, SCICR[ILT], determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

Note: With the WAKE bit clear, setting the SCICR[RWU] bit after UART_RXD has been idle can cause the receiver to wake up immediately.

20.2.7.2 Address Mark Wake-Up (WAKE = 1)

In address mark wake-up, a logic 1 in the MSB position of a frame clears the SCICR[RWU] bit and wakes up the SCI. This frame is considered to contain an address character. Hence, all data characters should have their MSB at zero. Each receiver software evaluates the addressing information when awakened and compares it to its own address. If the addresses match, the receiver(s) process the frames that follow. If the addresses do not match, the receiver software puts the receiver to sleep by setting the SCICR[RWU] bit. The RWU bit remains set and the receiver remains on standby until another address frame appears on UART_RXD.

The logic 1 in the MSB of an address character clears the receiver RWU bit before the stop bit is received and sets the SCISR[RDRF] interrupt flag. Address mark wake-up allows messages to contain idle characters but requires that the MSB be reserved for use in address frames.

20.3 Reset Initialization

After reset the UART transmitter and receiver are disabled and UART_TXD and UART_RXD are not driven. For information on initializing the transmitter, refer to **Section 20.1.1**. For information on initializing the receiver, refer to **Section 20.2.1**.

20.4 Modes of Operation

The following sections summarize the UART modes of operation.

20.4.1 Run Mode

Run mode is the normal mode of operation.

20.4.2 Single-Wire Operation

Normally, the UART (SCI) uses two signals for transmitting and receiving data. In single-wire operation, UART_RXD is disconnected from the UART and is available as a GPIO signal (see **Chapter 22**, *GPIO*). The UART uses UART_TXD for both receiving and transmitting data. Setting the data direction bit for UART_TXD, SCIDDR[DDRTX], configures UART_TXD as the output for transmitted data. Clearing the data direction bit, SCIDDR[DDRTX], disables the transmitter to drive UART_TXD.

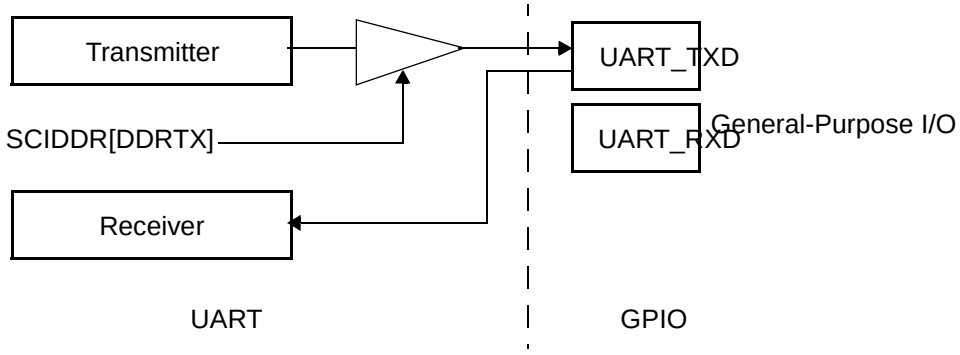


Figure 20-19. Single-Wire Operation (LOOPS = 1, RSRC = 1)

Enable single-wire operation by setting the SCICR[LOOPS] bit and the receiver source bit, SCICR[RSRC]. Setting the SCICR[LOOPS] bit disables the path from UART_RXD to the receiver. Setting the SCICR[RSRC] bit connects the receiver input to the output of UART_TXD. Both the transmitter and receiver must be enabled (SCICR[TE] = 1 and SCICR[RE] = 1). You can configure UART_TXD (see **Chapter 22**, *GPIO*) for full CMOS drive or for open-drain drive. The configuration bit controls UART_TXD in both normal operation and single-wire operation. The configuration bit also allows the UART_TXD outputs to be tied together in a multiple-transmitter system, which allows the UART_TXD signals of nonactive transmitters to follow the logic level of an active one. External pull-up resistors are necessary when using open-drain outputs.

20.4.3 Loop Operation

To help isolate system problems, the Loop mode is sometimes used to check software without changing the physical connections in the external system. In Loop mode, the transmitter output is connected to the receiver input internally. The UART_RXD signal is disconnected from the external connection which then becomes available for use as a GPIO signal. Clearing the data direction bit of UART_TXD disconnects the transmitter output from the external connection.

To enable loop operation, set the SCICR[LOOPS] bit and clear SCICR[RSRC]. Setting the SCICR[LOOPS] bit disables the path from UART_RXD to the external output connection. Clearing the SCICR[RSRC] bit connects the transmitter output to the receiver input. Both the transmitter and receiver must be enabled (SCICR[TE] = 1 and SCICR[RE] = 1) for loop operation.

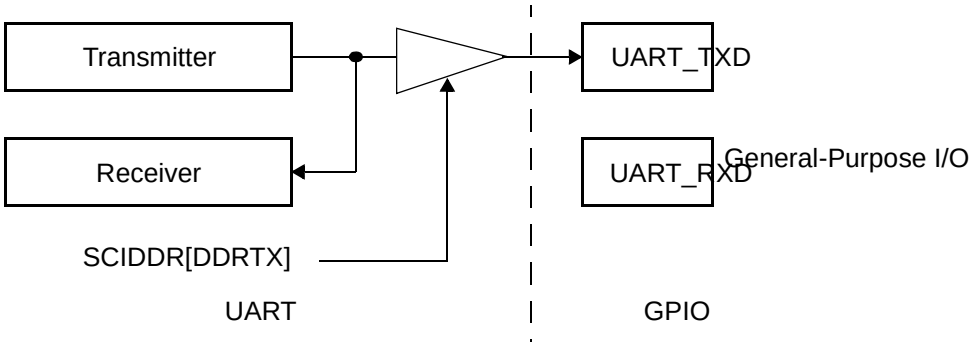


Figure 20-20. Loop Operation (LOOPS = 1, RSRC = 0)

20.4.4 Stop Mode

The UART stops its clock to provide reduced power consumption when the GCR1[UART_STC] bit is set (see **Section 8.2.1**, *General Configuration Register 1 (GCR1)*, on page 8-2). When the UART enters Stop mode, the states of the UART registers are unaffected. The UART registers cannot be accessed during Stop mode. When the UART_STC bit is cleared, UART operation resumes. Entering Stop mode during a transmission or reception results in invalid data. Therefore, disable the receiver and transmitter (SCICR[TE] = 0, SCICR[RE] = 0) before entering Stop mode.

20.4.5 Receiver Standby Mode

Refer to **Section 21.2.7**, *Receiver Wake-Up*.

20.5 Interrupt Operation

Table 20-7 lists the five interrupts generated by the UART to communicate with an SC3850 core or external host.The UART outputs only one signal, which can be activated by each of the five interrupt sources (refer to **Figure 20-1**, *UART Interface*, on page 20-1). Receiver interrupts are disabled when the receiver is in standby state (RWU is set).

Table 20-7. UART Interrupt Sources

Source	Transmitter/ Receiver	Interrupt Enable Bit	Flag at Status Register	Description
TDRE	T	TIE:SCICR[7]	TDRE:SCISR[15]	Indicates that a character was transferred from SCIDR to the transmit shift register.
TC	T	TCIE:SCICR[6]	TC:SCISR[14]	Indicates that a transmit is complete.
RDRF	R	RIE:SCICR[5]	RDRF:SCISR[13]	Indicates that received data is available in SCIDR.
OR	R	RIE:SCICR[5]	OR:SCISR[11]	Indicates an overrun condition.
IDLE	R	ILIE:SCICR[4]	IDLE:SCISR[12]	Indicates that receiver input has become idle.
Note: For details, refer to SCI Status Register (SCISR), on **page 20-29**				

The UART (SCI) only originates interrupt requests. An interrupt source flag (see **Table 20-7**) generates interrupt request if its associated interrupt enable bit is set. The interrupt vector offset and interrupt number are chip dependent.

20.6 UART Programming Model

All UART registers are mapped into the MBus address space. This section describes the UART (SCI) module registers, which are listed as follows:

- SCI Baud-Rate Register (SCIBR), on **page 20-25**.
- SCI Control Register (SCICR), on **page 20-26**.
- SCI Status Register (SCISR), on **page 20-29**.
- SCI Data Register (SCIDR), on **page 20-31**.
- SCI Data Direction Register (SCIDDR), on **page 20-32**.

Note: The UART register use a base address of: 0xFFF26C00.

20.6.1 SCI Baud-Rate Register (SCIBR)

SCIBR SCI Baud-Rate Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	—			SBR12	SBR11	SBR10	SBR9	SBR8	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SCIBR determines the SCI baud rate. A write to SCIBR[12–8] has no effect without a write to SCIBR[7–0], since writing to SCIBR[12–8] puts the data in a temporary location until SCIBR[7–0] is written.

Note: The formula for calculating the baud rate is: SCI baud rate = (CLASS clock/2)/(16 × BR).

Note: The baud-rate generator is disabled until the SCICR[TE] bit or the SCICR[RE] bit is set for the first time after reset. The baud-rate generator is disabled when BR = 0.

Table 20-8. SCIBR Bit Descriptions

Name	Reset	Description	Settings
— 31-13	0	Reserved. Write to zero for future compatibility.	
SBR[12–0] 12-0	4	**SCI Baud Rate** The baud-rate register used by the counter to determine the baud rate of the SCI.	Can contain a value from 1 to 8191.

20.6.2 SCI Control Register (SCICR)

SCICR SCI Control Register Offset 0x08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	LOOPS	—	RSRC	M	WAKE	ILT	PE	PT	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-9. SCICR Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
LOOPS 16	0	**Loop Select Bit** Disables the path from UART_RXD to the receiver input for loop (RSRC = 0) or single-wire mode (RSRC =1). See **Table 20-10**. The transmitter and the receiver must be enabled to use the loop functions. The receiver input is determined by the RSRC bit. The transmitter output is controlled by SCIDDR[DDRTX] bit. If the data direction bit (SCIDDR[DDRTX]) for UART_TXD is set and LOOPS = 1, the transmitter output drives UART_TXD. If the data direction bit is clear and LOOPS =1, the SCI transmitter does not drive UART_TXD.	1 Loop operation enabled. 0 Normal operation enabled.
— 14	0	Reserved. Write to zero for future compatibility.	
RSRC 13	0	**Receiver Source Bit** When LOOPS = 1, determines the internal feedback path for the receiver.	1 Receiver input connects to UART_TXD. 0 Receiver input internally connected to transmitter output.
M 12	0	**Data Format Mode Bit** Determines whether data characters are eight or nine bits long.	1 One start bit, nine data bits, one stop bit. 0 One start bit, eight data bits, one stop bit.
WAKE 11	0	**Wake** Determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received data character or an idle condition on UART_RXD (10 consecutive logic 1s if M = 0 or 11 consecutive logic 1s if M=1).	1 Address mark wake-up. 0 Idle line wake-up.

Table 20-9. SCICR Bit Descriptions (Continued)

Name	Reset	Description	Settings
ILT 10	0	**Idle Line Type Bit** Determines when the receiver starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.	1 Idle character bit count begins after stop bit. 0 Idle character bit count begins after start bit.
PE 9	0	**Parity Enable Bit** Enables the parity function. When enabled, the parity function inserts (when transmitter enabled) and checks (when receiver enabled) a parity bit at the most significant bit position.	1 Parity function enabled. 0 Parity function disabled.
PT 8	0	**Parity Type Bit** Determines whether the SCI generates and checks for even parity or odd parity. With even parity, an even number of 1s clears the parity bit and an odd number of 1s sets the parity bit. With odd parity, an odd number of 1s clears the parity bit and an even number of 1s sets the parity bit.	1 Odd parity. 0 Even parity.
TIE 7	0	**Transmitter Interrupt Enable** Enables the transmit data register empty flag, TDRE, to generate interrupt requests. **Note:** Since SCISR[TDRE] reset value is 1, setting TIE immediately after reset results in a UART interrupt request, regardless of SCICR[TE].	1 TDRE interrupt source enabled. 0 TDRE interrupt source disabled.
TCIE 6	0	**Transmission Complete Interrupt Enable** Enables the transmission complete flag, TC, to generate interrupt requests. **Note:** Since the SCISR[TC] reset value is 1, setting TCIE immediately after reset results in a UART interrupt request, regardless of SCICR[TE].	1 TC interrupt source enabled. 0 TC interrupt source disabled.
RIE 5	0	**Receiver Full Interrupt Enable** Enables the receive data register full flag, RDRF, and the overrun flag, OR, to generate interrupt requests.	1 RDRF and OR interrupt sources enabled. 0 RDRF and OR interrupt sources disabled.
ILIE 4	0	**Idle Line Interrupt Enable** Enables the idle line flag, IDLE, to generate interrupt requests.	1 IDLE interrupt source enabled. 0 IDLE interrupt source disabled.
TE 3	0	**Transmitter Enable** Enables the SCI transmitter. The TE bit can be used to queue an idle preamble.	1 Transmitter enabled. 0 Transmitter disabled.
RE 2	0	**Receiver Enable** Enables the SCI receiver.	1 Receiver enabled. 0 Receiver disabled.

Table 20-9. SCICR Bit Descriptions (Continued)

Name	Reset	Description	Settings
RWU 1	0	**Receiver Wake-Up** Enables the wake-up function and inhibits further receiver interrupt requests. Normally, hardware wakes the receiver by automatically clearing RWU.	1 RWU, Standby state. 0 Normal operation.
SBK 0	0	**Send Break** Toggling this bit sends one break character (10 or 11 logic 0s). Toggling implies clearing the SBK bit before the break character has finished transmitting. As long as SBK is set, the transmitter continues to send break characters.	1 Transmit break characters. 0 No break characters.

Table 20-10. Loop Functions

LOOPS	RSRC	SCIDDR[DDRTX]	Function
0	x	x	Normal operation
1	0	0	Loop mode, UART_TXD is not driven by the SCI transmitter
1	0	1	Loop mode, UART_TXD is driven by the SCI transmitter
1	1	0	Single-wire mode UART_TXD acting as an input for the received data. The external connection that UART_RXD shares can be configured as a GPIO.
1	1	1	Single-wire mode with UART_TXD acting as an output for the transmitted data. The transmitted data is also internally connected to the receiver input. The external connection that UART_RXD shares can be configured as a GPIO.
Note: See **Chapter 22**, *GPIO* for details on configuring the signal multiplexing.			

20.6.3 SCI Status Register (SCISR)

SCISR SCI Status Register Offset 0x10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	—							RAF
Type	R								R/W							R
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SCISR can be read any time. A write has no meaning or effect.

Table 20-11. SCISR Bit Descriptions

Name	Reset	Description	Settings
— 31–16	0	Reserved. Write to zero for future compatibility.	
TDRE 15	1	**Transmit Data Register Empty Flag** Set when the transmit shift register receives a character from the SCI data register. When TDRE is 1, the transmit data register (SCIDR) is empty and can receive a new value to transmit. This flag can generate an interrupt request (refer to **Section 20.5**). Clear TDRE by reading TDRE and then writing to T[7–0] in the SCIDR.	1 Character transferred to transmit shift register; transmit data register empty. 0 No character transferred to transmit shift register.
TC 14	1	**Transmit Complete Flag** Set low when there is a transmission in progress or when a preamble or break character is loaded. TC is set high when the TDRE flag is set and no data, preamble, or break character is being transmitted. When TC is set, UART_TXD becomes idle (logic 1). This flag can generate an interrupt request (refer to **Section 20.5**). Clear TC by reading TC and then writing to T[7–0] in the SCIDR. TC is cleared automatically when data, preamble, or break is queued and ready to be sent. Also, TC is cleared in the event of a simultaneous set and clear of the TC flag (transmission not complete).	1 No transmission in progress. 0 Transmission in progress.
RDRF 13	0	**Receive Data Register Full Flag** Set when the data in the receive shift register transfers to the SCI data register. This flag can generate an interrupt request (refer to **Section 20.5**). Clear RDRF by reading RDRF bit at SCISR and then reading R[7–0] in the SCIDR. **Note:** Once the RDRF flag is cleared, after it is set by a break or idle character, a valid frame must set the RDRF flag before another break or idle character can set it again.	1 Received data available in SCI data register. 0 Data not available in SCI data register.

Table 20-11. SCISR Bit Descriptions (Continued)

Name	Reset	Description	Settings
IDLE 12	0	**Idle Line Flag** Set when 10 consecutive logic 1s (if M = 0) or 11 consecutive logic 1s (if M = 1) appear on the receiver input. Once the IDLE flag is cleared, a valid frame must again set the RDRF flag before an idle condition can set the IDLE flag.This flag can generate an interrupt request (refer to **Section 20.5**). Clear IDLE by reading IDLE and then reading R[7–0] in the SCIDR. **Note:** When the receiver wake-up bit (RWU) is set, an idle line condition does not set the IDLE flag.	1 Receiver input has become idle. 0 Receiver input is either active now or has never become active since the IDLE flag was last cleared.
OR 11	0	**Overrun Flag** Set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. This flag can generate an interrupt request (refer to **Section 20.5**). Clear OR by reading OR then reading R[7–0] in the SCIDR.	1 Overrun. 0 No overrun.
NF 10	0	**Noise Flag** Set when the SCI detects noise on the receiver input. NF is set during the same cycle as the RDRF flag but is not set for an overrun. Clear NF by reading NF and then reading R[7–0] in the SCIDR.	1 Noise. 0 No noise.
FE 9	0	**Framing Error Flag** Set when a logic 0 is accepted as the stop bit. FE is set during the same cycle as the RDRF flag but is not set for an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading FE and then reading R[7–0] in the SCIDR.	1 Framing error. 0 No framing error.
PF 8	0	**Parity Error Flag** Set when the parity enable bit, PE, is set and the parity of the received data does not match its parity bit. Clear PF by reading PF and then reading R[7–0] in the SCIDR.	1 Parity error. 0 No parity error.
— 7–1	0	Reserved. Write to zero for future compatibility.	
RAF 0	0	**Receiver Active Flag** Set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RAF is cleared when the receiver detects an idle character.	1 Reception in progress. 0 No reception in progress.

20.6.4 SCI Data Register (SCIDR)

SCIDR SCI Data Register Offset 0x18

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R/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
	R8	T8	—						R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
Type	R	R/W	R/W						R[7–0] is R T[7–0] is W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note: In the SCIDR, writing affects only T[8–0]; writing to R[8–0] has no effect.

Table 20-12. SCIDR Bit Descriptions

Name		Reset	Description	Settings
— 31–16		0	Reserved. Write to zero for future compatibility.	
R8 15		0	**Received Bit 8** The ninth data bit received when the SCI is configured for 9-bit data format (M = 1).	
T8 14		0	**Transmit Bit 8** The ninth data bit transmitted when the SCI is configured for 9-bit data format (M = 1).	
— 13–8		0	Reserved. Write to zero for future compatibility.	
7–0	**R[7–0]**	0	**Received Bits 7–0** Received bits seven through zero for 9-bit or 8-bit data formats.	
	T[7–0]		**Transmit Bits 7–0** Transmit bits seven through zero for 9-bit or 8-bit formats.	

Notes:
1. If the value of T8 is the same as in the previous transmission, T8 does not have to be rewritten. The same value is transmitted until T8 is rewritten.
2. In 8-bit data format, only SCIDR[7–9] need to be accessed.
3. When transmitting in 9-bit data format, write to SCIDR[15–0] (one access). Otherwise, write first to T8 and then to the low byte (SCIDR[7–0]).

20.6.5 SCI Data Direction Register (SCIDDR)

SCIDDR SCI Data Direction Register Offset 0x28

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

When LOOPS is cleared and TE is set, UART_TXD is an output regardless of the state of SCIDDR[DDRTX].

Table 20-13. SCIDDR Bit Descriptions

Name	Reset	Description	Settings
— 31–10	0	Reserved. Write to zero for future compatibility.	
DDRTX 9	0	**Data Direction Bit TX** Controls the TX signal direction in single-wire mode (refer to **Section 20.4.2**).	1 If TE=1, TX is driven by the transmitter. Otherwise, if TE=0, UART_TXD is driven by logic 0. 0 UART_TXD is not driven when the transmitter is disabled (TE=0) or when LOOPS=1.
— 8–0	0	Reserved. Write to zero for future compatibility.	
Note: The setting descriptions assume that the UART_TXD signal is configured for UART operation.			

Timers 21

The MSC8156E device includes 3 types of timers:

- *Device-level timers*. Each MSC8156E device contains a total of 16 device timers. Each can be used by any of the DSP cores within MSC8156E as well as by an external host. See **Section 21.1** for details.
- *SC3850 DSP core subsystem timers*. Each of the SC3850 DSP core subsystems contain two timers used by the specific DSP core for any required operating system purpose. For details, see the *MSC8156 SC3850 DSP Core Subsystem Reference Manual* available with a signed non-disclosure agreement. Contact your Freescale representative or distributor for details.
- *Software watchdog timers.* The MSC8156E device includes 8 software watchdog timers. Each of the software watchdog timers can be used by any of the cores within MSC8156E as well as by an external host. For details, see **Section 21.3**.

21.1 Device-Level Timers

There are four identical quad timer modules in the MSC8156E device. Each quad timer module contains four identical timer groups that serve as frequency dividers, clock generators, and event counters. Each 16-bit timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, and two status and control registers. The timers interface with the MSC8156E inputs/outputs as listed in **Table 21-1**. This document uses the term *timer* to refer to the four internal timers in each module and quad timer to refer to each of the timer modules. The term channel is used in register naming for clarity.

Table 21-1. Device-Level Timers Connectivity

Timer Module	Timer Channel	External Input	External Output
0	0	TMR0	—
	1	TMR1	TMR0
	2	CLKIN	—
	3	TDM0TCK	TMR1
1	0	TMR0	—
	1	TMR2	—
	2	CLKIN	—
	3	TDM0TCK	TMR2

Table 21-1. Device-Level Timers Connectivity (Continued)

Timer Module	Timer Channel	External Input	External Output
2	0	TMR0	—
	1	TMR3	—
	2	CLKIN	—
	3	TDM0TCK	TMR3
3	0	TMR0	—
	1	TMR4	—
	2	CLKIN	—
	3	TDM0TCK	TMR4

Note: The external inputs list the external signal line connected to the specified timer input. The external outputs connect to the specified timer output. Most of the timer outputs do not connect to an external signal line. Even for cases in which a connection is indicated, the output is not valid unless the output is enabled by the TMRnSCTL[OEN] bit for the timer (see **Section 21.4.1.2**). Inputs and outputs for the TMRn signal lines are multiplexed.

21.1.1 Features

Features of the timers include:

- Counters support the following operations:
  - Cascade.
  - Preloading.
  - Count once or continuously.
  - Share input pins.
  - Do capture and compare.
  - Count up or down.
- Count modulo is programmable.
- Maximum count rate is the CLASS clock/2 rate when the timer input signals are not in use.
- Maximum count rate is half the CLASS clock/2 rate when the timer input signals are in use.
- Each counter has a separate prescaler.

21.1.2 Timer Module Architecture

Each quad timer module contains four timers. The block diagram of one timer within a quad timer module is shown in **Figure 21-1**. As the figure shows, the primary clock selector contains a prescaler, primary clock multiplex, and an optional invert. It selects and optionally inverts a clock source for the primary clock. The primary clock can be selected from any of the following:

- Normal clocking:
  - CLASS clock/2
  - (CLASS clock/2) divided by the prescaler: /1, /2, /4, ..., /128.
- Clocking from external events through a timer input signal.

- Clocking in Cascaded mode using an output from another timer in the same quad timer module.

Before a timer is enabled to count events, initialize the timer output signal by writing the desired initialization value to TMRxSCTL[VAL] (page 21-19) and then set the TMRxSCTL[FORC] bit (page 21-19).

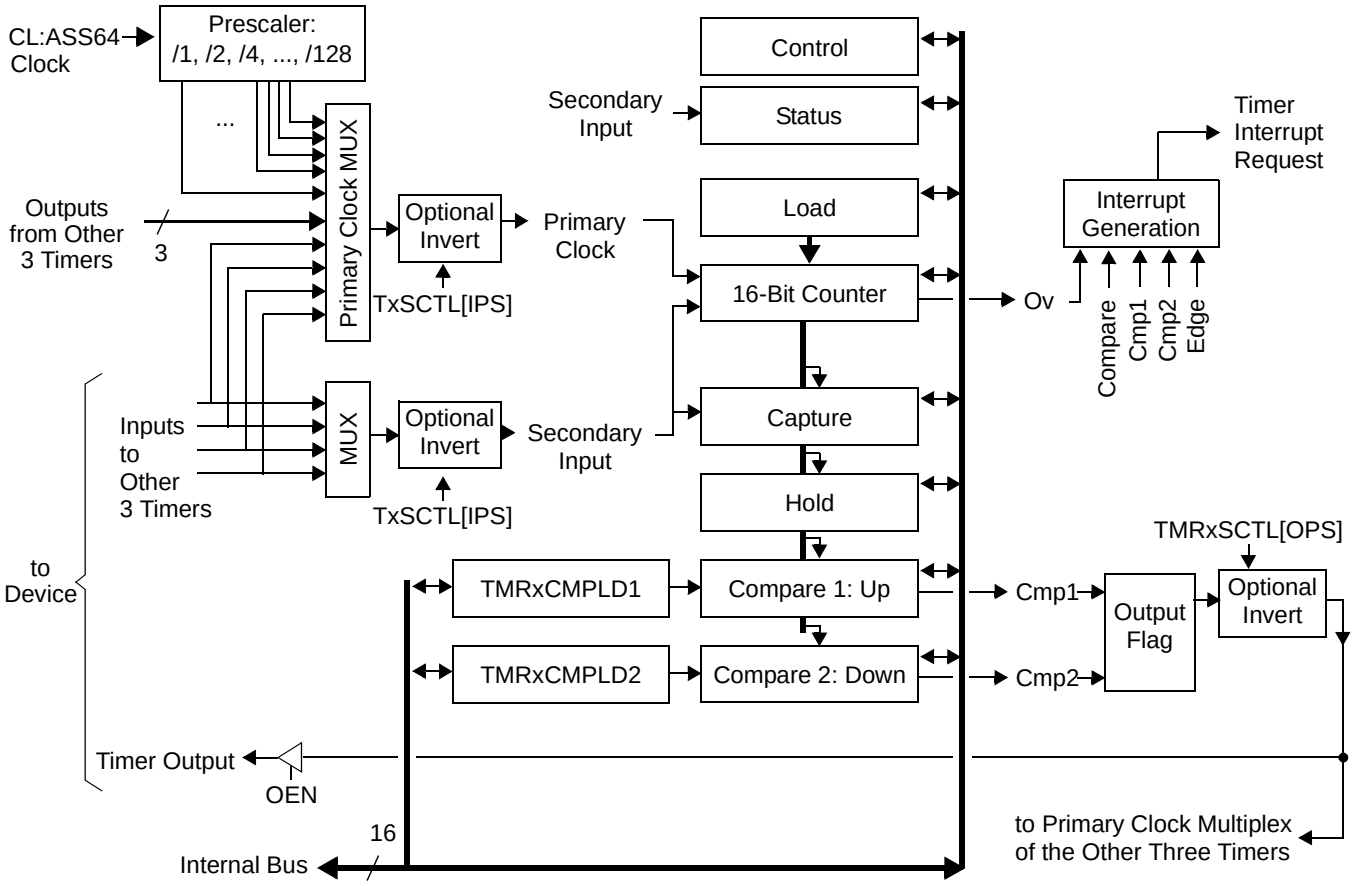


Figure 21-1. Timer Module Block Diagram — One of Four Timers

21.1.3 Setting Up Counters for Cascaded Operation

To create a counter larger than 16 bits, the first timer is programmed for the desired configuration and count mode (it is *not* programmed in Cascade mode). This first timer also programs the TMRxCTL[PCS] field to select the desired primary clock. All other timers in the cascade are simply programmed with their count mode set to Cascade mode (TMRxCTL[CM] = 111). They also program the TMRxCTL[PCS] field using the appropriate timer N output so that each can receive their clocks from the previous timer in the chain. The first timer in the chain must never be programmed in cascade mode. Also, the first timer in the chain must not choose one of the outputs from the timers for its primary clock. All timers in the cascade follow the counting mode of the first timer in the chain. In Cascade mode, a special high-speed signal path is used, bypassing the timer output flag logic to ensure that the cascaded channels operate as a single synchronous counter. You can connect timers using the other (non-cascade) timer modes and

selecting the outputs of other timers as a clock source. In this case, the timers operate in a *ripple* mode, in which higher-order counters transition a clock later than in a purely synchronous design. This is *not* the typical use for cascaded counters.

21.1.3.1 Operation of the Cascaded Timer

If the first timer in a cascaded chain is counting up and it encounters a compare event, the timer connected to it is incremented. If the first timer in the chain is counting down and it encounters a compare event, the timer is decremented. You can correctly read all 16-bit portions of a cascaded timer as follows using the TMRxHOLD registers:

1. Read any 16-bit portion of the cascaded timer from its TMRxCNTR register. You can do this at any time.
2. When any TMRxCNTR register in the module is read, all other timers simultaneously load their values into their hold registers.
3. Read the 16-bit portions of all other timers in the cascade from their TMRxHOLD registers.

21.1.3.2 Cascading Restrictions

To ensure that there are no feedback loops in a cascade, there are restrictions on which timers can be cascaded. The timer with the lowest number must always be the first in the cascade, the timer with the second lowest number must be second, and so on. The timer with the highest number must always be last in the cascade. **Table 21-2** summarizes the cascading restrictions.

Table 21-2. Restrictions On Cascading Timers

Timer Number	Valid Cascade Inputs	Legal Values for Cascading using TMRxCTL[PCS]	Description
Timer 0	None	None	Timer 0 can only be the first timer in a cascaded timer. It cannot receive another timer's output for cascaded operation. Timer 0 must always be the first timer in the cascade.
Timer 1	Timer 0 output	0100: Timer 0	Timer 1 can be cascaded with Timer 0, with Timer 0 as the first timer in the chain.
Timer 2	Timer 0 output Timer 1 output	0100: Timer 0 0101: Timer 1	Timer 2 can be cascaded with Timer 0 or Timer 1 when Timer 2 is not the first timer in the cascade.
Timer 3	Timer 0 output Timer 1 output Timer 2 output	0100: Timer 0 0101: Timer 1 0110: Timer 2	Timer 3 can be cascaded with Timer 0, Timer 1, or Timer 3. Timer 3 must always be the last timer in a cascade.

21.1.4 Timer Operating Modes

The timer operates in two modes:

- Count the CLASS clock/2 or external events via the timer input using the primary clock.
- Count the CLASS clock/2 or external events via the timer input using the primary clock while a second input signal, the secondary clock, is asserted, thus timing the width of the secondary clock signal.

Each timer can be configured in the following ways:

- to count the rising, falling, or both edges of the selected input pin.
- to decode and count quadrature encoded input signals.
- to count up and down using dual inputs in a count with direction format.
- program the timer terminal count value (modulo).
- program the value loaded into the timer after it reaches its terminal count.
- program the timer to count repeatedly or to stop after completing one count cycle.
- program the timer to count to a programmed value (using the compare functionality) and then immediately reinitialize or to count through the compare value until the count rolls over to zero.

The counting modes define the different modes for clocking the timers. The count mode is selected in the TMRxCTL[CM] field (page 21-17). If a timer is programmed to count to a specific value and then stop, the TMRCTL[CM] bit is cleared when the count terminates. **Table 21-3** summarizes the counting modes.

Table 21-3. Summary of Timer Counting Modes

Counting Mode	CM Bits	Description	Primary Clock	Secondary Clock
Disabled	000	Timer not active.	—	—
Count	001	Counts the rising edges of the selected clock source (falling edges if TxSCTL[IPS] is set). This mode is useful for generating periodic interrupts for timing purposes or for counting external events.	Clock*	—
Dual-Edge Count	010	Counts both edges of a timer Input signal. This mode is useful for counting the changes in the external environment. When this mode is selected, TMRxCTL[PCS] must not be set to any value between 1000 and 1111; that is, it must not set to the input clock or any scaled version of the input clock.	Clock	—

Table 21-3. Summary of Timer Counting Modes

Counting Mode	CM Bits	Description	Primary Clock	Secondary Clock
Gated Count	011	Counts primary clock edges while the secondary input is high (low if TxSCTL[IPS] is set). This mode is used to time the duration of external events when the primary clock is set to the input clock and the secondary input is set to use one of the timer input signals. It can also be used to count the number of external events that occur on one of the timer input signals, set as the primary clock, while a second timer input signal, connected to the secondary Input signal, is asserted.	Clock	Gate*
Quadrature Count	100	Counts using quadrature encoded signals. The quadrature signals are square waves, 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information. A timing diagram illustrating the basic operation of a quadrature incremental position encoder is provided in **Figure 21-2.**	Quadrature signal	Quadrature signal
Signed Count	101	Counts the primary clock source while a secondary input provides the count direction (up or down) for each recognized count.	Clock to count	Count direction
Triggered Count	110	Counts the primary clock source only after a rising edge is detected on the secondary input (falling edge if TxSCTL[IPS] is set). The counting continues until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count is reached, counting stops. Subsequent odd-numbered edges of the secondary input restart the counting, and even numbered edges stop counting. This process continues until a compare event occurs.	Clock to count	Enable/disable timer*
Cascade Count	111	Cascades multiple timers. Cascade mode is used for creating timers larger than 16-bits. Up to four timers may be cascaded together to create a 64-bit wide timer. The Cascaded Timer mode is synchronous. See **Section 21.1.4.2**.	Clock to count	Triggers timer
Note: * This input can be inverted by the TxSCTL[IPS] bit.				

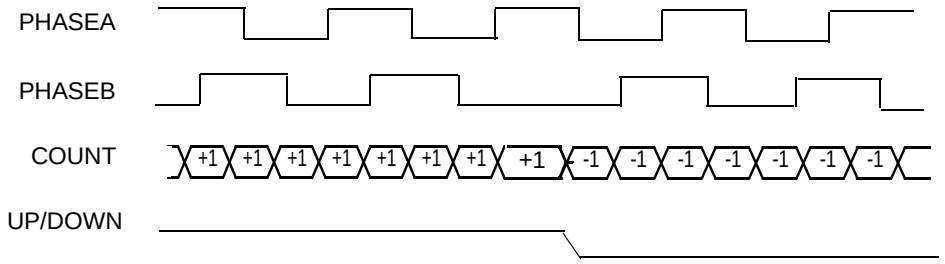


Figure 21-2. Quadrature Incremental Position Encoder

Other count modes derived as special cases of the modes described in **Table 21-3** are described in the remainder of this section.

21.1.4.1 One-Shot Mode

One-Shot mode is a variation on Triggered Count mode if the timer is set up as follows:

- TMRxCTL[CM] = 110 to count the rising and falling edges of the primary source (see **Table 21-5**).
- The Count Length bit, TMRxCTL[LEN] = 1.
- Output Flag mode, TMRxCTL[OFLM] = 101 to select set on compare, cleared on secondary input signal edge.
- The Count Once bit, TMRxCTL[ONCE] = 1 to count till a compare and then stop.

An external event causes the timer to count. When terminal count is reached, the timer output flag is asserted. This delayed output assertion can be used to provide timing delays.

21.1.4.2 Pulse Output Mode

In Pulse Output mode, a variation on Count mode, the timer outputs a stream of pulses with the same frequency as the selected clock source (cannot be the (CLASS clock/2)/1) if the timer is set up as follows:

- TMRxCTL[CM] = 001 to count the rising edges of the primary source.
  (see **Table 21-5*TMR[0–3]SCTL[0–3] Bit Descriptions,*** on page 21>-19).
- The Output Flag Mode, TMRxCTL[OFLM], = 111 to enable gated clock output while the timer is active.
- The Count Once bit, TMRxCTL[ONCE], = 1 to count till a compare and then stop.

The number of output pulses is equal to the compare value minus the initial value. The primary count source must be set to one of the timer outputs for gated clock output mode.

21.1.4.3 Fixed Frequency PWM Mode

Fixed Frequency Pulse Width Modulated (PWM) mode is a subset of Count mode. The timer is set up as follows:

- TMRxCTL[CM] = 001 to count the rising edges of the primary source.
- The Count Length bit, TMRxCTL[LEN], = 0 so that the timer continues counting past the compare value (binary roll-over).
- The Count Once bit, TMRxCTL[ONCE], = 0 to count repeatedly.
- The Output Flag Mode, TMRxCTL[OFLM], = 110 so that the output flag is set when a compare occurs.

The timer output yields a PWM signal with:

- a frequency equal to the count clock frequency divided by 65,536.
- a pulse width duty cycle equal to the compare value divided by 65,536.

21.1.4.4 Variable Frequency PWM Mode

The timer output yields a PWM signal with a frequency and pulse width determined by the values programmed into the TMRxCMP1 and TMRxCMP2 registers and the input clock frequency if the timer is set up as follows:

- TMRxCTL[CM] = 001 TMRxCTL[CM] = 001 to count the rising edges of the primary source (see **Table 21-5**).
- The Count Length bit, TMRxCTL[LEN], = 1 so that the timer counts to the compare value and then reinitializes.
- The Count Once bit, TMRxCTL[ONCE], = 0 to count repeatedly.
- The Output Flag Mode, TMRxCTL[OFLM], = 100 to toggle the timer output flag using alternating compare registers.

This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. The TMRxCMPLD1 and TMRxCMPLD2 registers are especially useful for this mode because they give you time to calculate values for the next PWM cycle during the PWM current cycle.

To set up the timer to run in Variable Frequency PWM mode with compare preload, use the set up described here for the desired timer. During set-up, update the TMRxCTL register last because the timer starts counting if the count mode changes to any value other than 000. Set up the Timer Control (TMRxCTL) register bits as follows:

- Count Mode (CM) = 001 to count the rising edges of the primary source.
- Primary Count Source (PCS) = 1000 to specify the best granularity for waveform timing; prescaler (CLASS clock/2)/1.
- Secondary Count Source (SCS) = Any value because the bits are ignored in this mode.
- Count Once (ONCE) = 0 to count repeatedly.
- Count Length (LEN) = 1 so that the timer counts till it reaches a compare and then reinitializes the timer register.
- Direction (DIR) = Count up (0) or count down (1). The compare register values must be chosen carefully to account for roll-under and so on.
- External Initialization (EIN) = 0 so that another timer cannot force a reinitialization of this timer. However, you can set this bit if you need the functionality.
- Output Mode (OFLM) = 100 to toggle the timer output flag using alternating compare registers.

Set up the Timer Status and Control Register (TMRxSCTL) bits as follows:

- Output Polarity Select (OPS) = Your choice, true (0) or inverted (1).
- Output Enable (OEN) = 1 to enable the timer output to be put on an external pin. Set this bit as needed.
- Ensure that the rest of the TMRxSCTL bits are cleared. Interrupts are enabled in the Timer Comparator Status and Control Register (TMRxCOMSC) instead of in this register.

Set up the Timer Comparator Status and Control Register (TMRxCOMSC) bits as follows:

- Timer Compare 2 Interrupt Enable (TCF2EN) = 1 to allow an interrupt to be issued when TCF2 is set).
- Timer Compare1 Interrupt Enable (TCF1EN) = 0 so that an interrupt cannot be issued when TCF1 is set.
- Timer Compare 1 Interrupt Source (TCF1) = 0 to clear the timer compare 1 interrupt source flag. This bit is set when a successful comparison of the timer and the TMRxCMP1 register occurs.
- Timer Compare 2 Interrupt Source (TCF2) = 0 to clear the timer compare 2 interrupt source flag. This bit is set when a successful comparison of the timer and the TMRxCMP2 register occurs.
- Compare Load Control 1 (CL1) = 10 to load the compare register when TCF2 is set.
- Compare Load Control 2 (CL2) = 01 to load the compare register when TCF1 is set.

To service the TCF2 interrupts generated by the Timer, the interrupt controller must be configured to enable the interrupts for the timer being used. Additionally, you must write an interrupt service routine to do at least the following:

- Clear the TCF2 and TCF1 flags.
- Calculate and write new values for TMRxCMPLD1[15–0] and TMRxCMPLD2[15–0].

Figure 21-3 shows the timing for the compare preload cycle, which begins when a compare event on TMRxCMP2 causes TCF2 to be set. TMRxCMP1 is loaded with the value in the TMRxCMPLD1 one internal bus clock later. In addition, the timer asserts an interrupt, and the interrupt service routine executes while both comparator load registers are updated with new values. When TCF1 is set, TMRxCMP2 is loaded with the value of the CLV2 bits in TMRxCMPLD2. During the subsequent TCF2 event, TMRxCMP1 is loaded with the value of the TMRxCMPLD1[CLV1] bits. The cycle starts over again as an interrupt is asserted and the interrupt service routine clears TCF1 and TCF2 and calculates new values for TMRxCMPLD1 and TMRxCMPLD2.

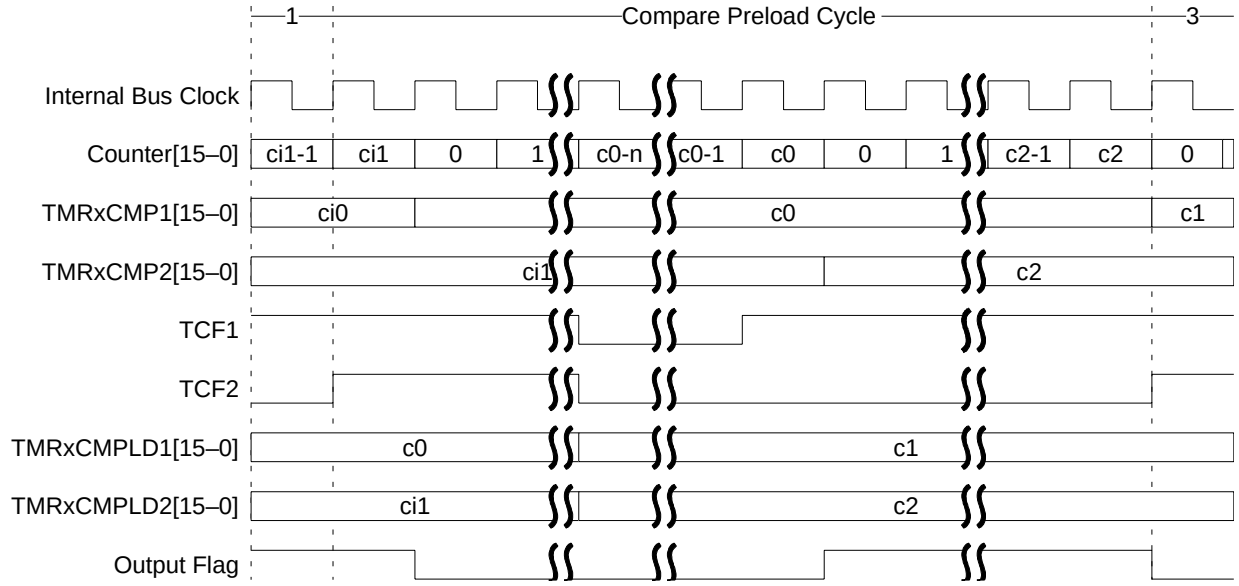


Figure 21-3. Compare Preload Timing

21.1.5 Timer Compare Functionality

The compare registers (TMRxCMP1 and TMRxCMP2) provide a bidirectional modulo count capability. TMRxCMP1 is used when the timer is counting *up*. Program it with the desired maximum count value or to 0xFFFF to indicate the maximum un signed value prior to roll-over. TMRxCMP2 is used when the timer is counting *down*. Program it with the maximum negative count value or to 0x0000 to indicate the minimum unsigned value prior to roll-under. The only exception occurs when the timer is operating with alternating compare registers.

When TMRxCTL[OFLM] = 100, alternating values of TMRxCMP1 and TMRxCMP2 are used to generate successful compares, and the output flag toggles while using alternating compare registers. For example, when TMRxCTL[OFLM] = 100, the timer is programmed to count upwards. It counts until the TMRxCMP1 value is reached, reinitializes, then counts until the TMRxCMP2 value is reached, reinitializes, then counts until the TMRxCMP1 value is reached, and so on. In this Variable Frequency PWM mode, the TMRxCMP2 value defines the desired pulse width of the *on-time*, and the TMRxCMP1 register defines the *off-time*. The Variable Frequency PWM mode is defined for positive counting only. See **Section 21.1.4.4**, *Variable Frequency PWM Mode*, on page 21-8.

Use caution when changing TMRxCMP1 and TMRxCMP2 while the timer is active. If the timer has already passed the new value, it counts to 0xFFFF or 0x0000, rolls over/under, and then begins counting toward the new value. The check is for Count = TMRxCMPx, not Count > = TMRxCMP1 or Count < = TMRxCMP2). Use of the preload registers addresses this problem.

21.1.5.1 Compare Preload Registers

The TMRxCMPLD1, TMRxCMPLD2 and TMRxCOMSC registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality, use the loading method described in this section.

The compare preload feature speeds updating of the compare registers. The compare preload feature allows you to calculate new compare values and store them into the comparator preload registers. When a compare event occurs, the new compare values in the comparator preload registers are directly written to the compare registers, eliminating the use of software to do this.

The compare preload feature is used in variable frequency PWM mode. See **Section 21.1.4.4**, *Variable Frequency PWM Mode*, on page 21-8. The TMRxCMP1 register determines the pulse width for the logic low part of the timer output, and TMRxCMP2 determines the pulse width for the logic high part of the timer output. The period of the waveform is determined by the TMRxCMP1 and TMRxCMP2 values and the frequency of the primary clock source.
See **Figure 21-4**.

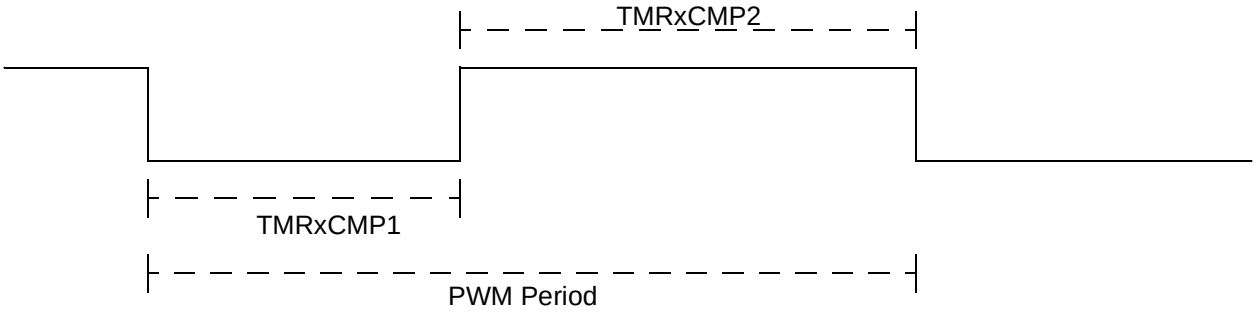


Figure 21-4. Variable PWM Waveform

To update the duty cycle or period of the waveform, update the TMRxCMP1 and TMRxCMP2 values using the compare preload feature.

21.1.5.2 Capture Register Use

The capture register, TMRxCAP (page 21-23), stores a copy of the timer value when an input edge (positive, negative, or both) on the secondary input signal is detected. The capture mode, programmable via TMRxSCTL[CM] (page 21-19), is one of the following:

- CM = 00. Disabled.
- CM = 01. Load the capture register on the *rising* edge of the signal.
- CM = 10. Load the capture register on the *falling* edge of the signal.
- CM = 11. Load the capture register on *either* edge of the signal.

When a capture event occurs, there are no further updates of TMRxCAP until the input edge flag (IEF) is cleared by writing a value of 0 to the TMRxSCTL[IEF] bit (page 21-19).

21.1.5.3 Broadcast from an Initiator Timer

Any timer can be assigned as an initiator. An initiator compares signal can be broadcast to the other timers within the module. The other timers can be configured as follows to reinitialize their timers and/or force their output to predetermined values when an initiator timer compare event occurs:

- Select one timer as the initiator timer by setting the TMRxSCTL[MSTR] bit.
- Program the other timers to perform an action when a compare event occurs on the initiator timer as follows:
  - The other timer is reinitialized if its TMRxCTL[EIN] bit is set.
  - The other timer forces its output flag signal if its TMRxSCTL[EEOF] bit is set.

21.1.6 Resets and Interrupts

The timers reset conditions are shown in **Chapter 5**, *Reset*. This reset forces all registers to their reset state and clears the output flag signal if it is asserted. The timer is turned off until the settings in the control register are changed. Each timer in a quad timer module can be programmed for interrupts. The available types of interrupts are as follows:

- Timer compare
- Timer compare 1
- Timer compare 2
- Timer overflow
- Timer input edge

Each of these different types is ORed together within each timer to generate a single interrupt request signal to the interrupt controller.

21.1.6.1 Timer Compare Interrupts

Interrupt requests are generated when a successful compare occurs between a timer and its compare registers while the Timer Compare Flag Interrupt Enable bit, TMRxSCTL[TCFIE], is set. These interrupt requests are cleared by writing a zero to the appropriate TMRxSCTL[TCF] bit. When a timer compare interrupt is set in the TMRxSCTL and the compare preload registers are available, one of the following two interrupts is also asserted:

- Timer compare 1 interrupt
- Timer compare 2 interrupt

Timer compare 1 interrupts are generated when a successful compare occurs between a timer and its TMRxCMP1 register while the Timer Compare 1 Interrupt Enable (TCF1EN) is set in the TMRxCOMSC register. These interrupts are cleared by writing a zero to the TMRxCOMSC[TCF1] bit. Timer compare 2 interrupts are generated when a successful compare

occurs between a timer and its TMRxCMP2 register while the Timer Compare 2 Interrupt Enable (TCF2EN) bit is set in the TMRxCOMSC register. These interrupts are cleared by writing a zero to the TCF2 bit in the TMRxCOMSC.

21.1.6.2 Timer Overflow Interrupts

Timer overflow interrupts are generated when a timer rolls over its maximum value while the TCFIE bit is set in the TMRxSCTL register. These interrupts are cleared by writing a zero to the Timer Overflow Flag (TOF) bit of the appropriate TMRxSCTL.

21.1.6.3 Timer Input Edge Interrupts

Timer input edge interrupts are generated by a transition of the input signal (either positive or negative, depending on TMRxSCTL[IPS] setting) while the Input Edge Flag Interrupt Enable (IEFIE) bit is set in the TMRxSCTL. These interrupts are cleared by writing a zero to the appropriate TMRxSCTL[IEF] bit.

21.2 SC3850 DSP Core Subsystem Timers

For a detailed description of the core subsystem timers, see the *MSC8156 SC3850 DSP Core Subsystem Reference Manual*.

21.3 Software Watchdog Timers

There are total of eight identical software watchdog timers (WDTs). Typically, one is used per core and the remainder are used for external hosts. However, you can allocate the WDTs in any manner to meet your system requirements.

The WDT is responsible for asserting a hardware reset or machine-check interrupt (MCP) if the software fails to service the software watchdog timer for a certain period of time (for example, because software is lost or trapped in a loop with no controlled exit). Each WDT is a free-running down-counter that generates a reset or a non-maskable interrupt on underflow. To prevent a reset, software must periodically restart the countdown. Watchdog timer operations are configured in the system watchdog control register (SWCRR). See **Section 21.4.3.1** for details.

Note: If any of the watchdog timers generate a reset, it resets all of the SC3850 core subsystems in the MSC8156E device.

The software watchdog timer is enabled after reset to cause a hard reset or non-maskable interrupt (MCP) if it times out. If the software WDT is not needed, you must clear SWCRR[SWEN] to disable it. If it is used, the software WDT requires a special service sequence that executes periodically. Without this periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt, as programmed in SWCRR[SWRI]. Once software writes SWRI, the state of SWEN cannot be changed. **Figure 21-1** shows the high level WDT block diagram.

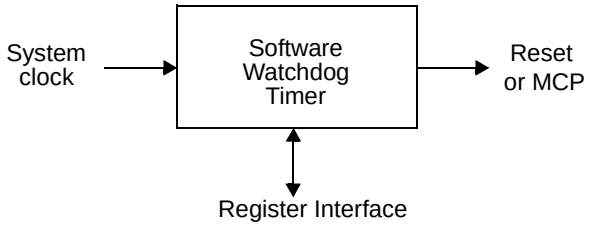


Figure 21-1. Software Watchdog Timer High Level Block Diagram

21.3.1 Features

The key features of the WDT include the following:

- Based on 16-bit prescaler and 16-bit down-counter.
- Provide a selectable range for the time-out period.
- Provide ~8.59 s maximum software time-out delay for 500 MHz input clock.

21.3.2 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

  If the software watchdog timer is not needed, user can disable it. SWCRR[SWEN] bit enables the watchdog timer. It should be cleared by software after a system reset to disable the software watchdog timer. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state.

  — WDT enable mode (SWCRR[SWEN] = 1)
  This is the default value after soft reset.

  — WDT disable mode (SWCRR[SWEN] = 0)
  If the software watchdog timer is not needed, the user must clear SWCRR[SWEN] to disable it.

- WDT reset/interrupt output mode

  Without software periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt (MCP), programmed in SWCRR[SWRI].

  According to SWCRR[SWRI] programming, WDT timer causes a hard reset or machine check interrupt to the core.

— Reset mode (SWCRR[SWRI] = 1)
Software watchdog timer causes a hard reset (this is the default value after soft reset).

— Interrupt mode (SWCRR[SWRI] = 0)
Software watchdog timer causes a machine check interrupt to the core.

- WDT prescaled/non-prescaled clock mode

  The WDT counter clock can be prescaled by programming CRR[SWPR] bit that controls the divide-by-65536 of WDT counter.

  — Prescale mode (CRR[SWPR] = 1)
  The WDT clock is prescaled.

  — Non-prescale mode (CRR[SWPR] = 0)
  The WDT clock is not prescaled.

21.3.3 Software WDT Servicing

The software watchdog timer service sequence consists of the following two steps:

- Write 0x556C to the System Watchdog Service Register (SWSRR)
- Write 0xAA39 to SWSRR

The service sequence reloads the watchdog timer and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the SWSRR, the entire sequence must start over. Although the writes must occur in the correct order before a time-out, any number of instructions can be executed between the writes. This allows interrupts and exceptions to occur between the two writes when necessary. **Figure 21-5** shows a state diagram for the watchdog timer.

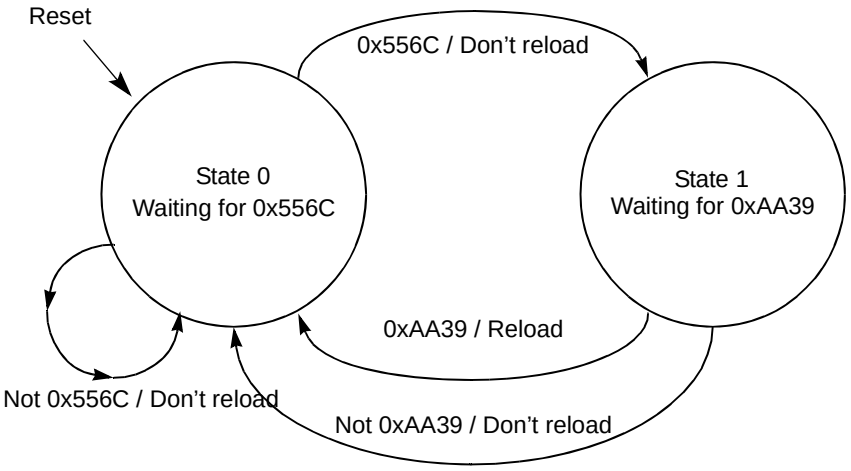


Figure 21-5. Software Watchdog Timer Service State Diagram

Although most software disciplines permit or even encourage the watchdog concept, some systems require a selection of time-out periods. For this reason, the software watchdog timer

must provide a selectable range for the time-out period. **Figure 21-6** shows how to handle this need.

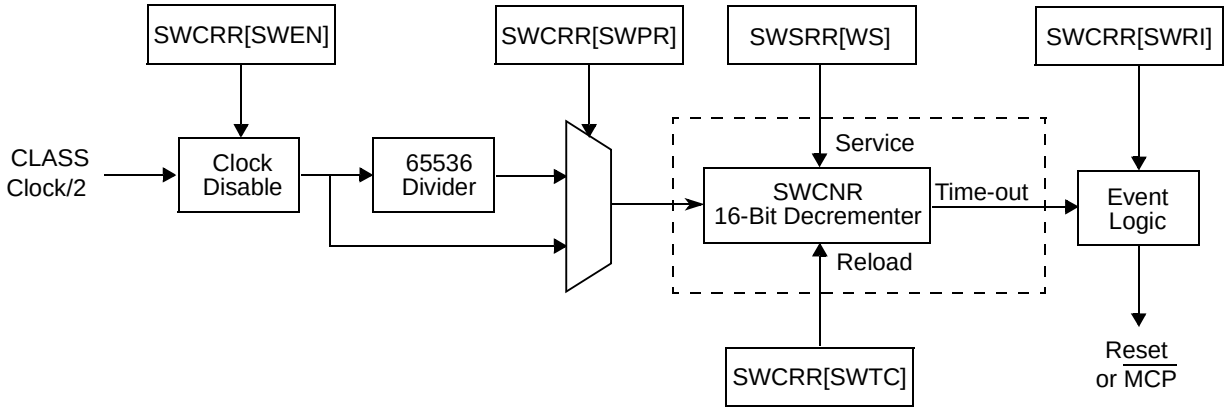


Figure 21-6. Software Watchdog Timer Functional Block Diagram

In **Figure 21-6**, the range is determined by SWCRR[SWTC]. The value in SWTC is then loaded into a 16-bit decrementer clocked by the CLASS clock/2. An additional divide-by-65536 prescaler value is used when needed.

The decrementer begins counting when loaded with a value from SWTC. After the timer reaches 0x0, a software watchdog expiration request is issued to the reset or MCP (machine check processor) control logic. Upon reset, SWTC is set to the maximum value and is again loaded into the System Watchdog Service Register (SWSRR), starting the process over. When a new value is loaded into SWTC, the software watchdog timer is not updated until the servicing sequence is written to the SWSRR. If SWCRR[SWEN] is loaded with 0, the modulus counter does not count.

21.4 Timers Programming Model

Because they are programmed differently, the device-level, SC3850 DSP core platform level, and software watchdog timers are described in separate subsections.

21.4.1 Device-Level Timers

This section describes the device-level timers. For a complete listing of all registers in all modules with their memory locations, see **Chapter 9**, *Memory Map*. The device-level timer registers are listed as follows, along with the pages on which the registers are discussed:

- Timer Channel Control Registers (TMR[0–3]CTL[0–3], page 21-17.
- Timer Channel Status and Control Registers (TMR[0–3]SCTL[0–3]), page 21-19.
- Timer Channel Compare Register 1 (TMR[0–3]CMP1[0–3]), page 21-21.
- Timer Channel Compare Register 2 (TMR[0–3]CMP2[0–3]), page 21-21.

- Timer Channel Compare Load Register 1 (TMR[0–3]CMPLD1[0–3]), page 21-22.
- Timer Channel Compare Load Register 2 (TMR[0–3]CMPLD2[0–3]), page 21-22.
- Timer Channel Comparator Status and Control Registers (TMR[0–3]COMSC[0–3]), page 21-22.
- Timer Channel Capture Register (TMR[0–3]CAP[0–3]), page 21-22.
- Timer Channel Load Register (TMR[0–3]LOAD[0–3]), page 21-24.
- Timer Channel Hold Registers (TMR[0–3]HOLD[0–3]), page 21-24.
- Timer Channel Counter Register (TMR[0–3]CNTR[0–3]), page 21-24.

Note: The base addresses for the device level timer modules are as follows:

- Timer 0 = 0xFFF26000
- Timer 1 = 0xFFF26100
- Timer 2 = 0xFFF26200
- Timer 3 =- 0xFFF26300

21.4.1.1 Timer Channel Control Registers (TMRnCTLx)

TMR[0–3]CTL[0–3] Timer Control Registers Offset 0x18 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CM			PCS				SC		ONCE	LEN	DIR	EIN	OFLM		
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-4. TMR[0–3]CTL[0–3] Bit Descriptions

Name	Reset	Description	Settings	
CM 15–13	0	**Count Mode** Control the basic counting behavior of the counter. Rising edges are counted only when TxSCTL[IPS] = 0. Falling edges are counted only when TxSCTL[IPS] = 1. When count mode 010 is selected, the PCS bits must not be set to 1000–1111. When count mode 111 is selected, the PCS bits must be set to one of the “Timer N output” selections.	000 001 010 011 100 101 110 111	No operation. Disabled. Count rising edges of the primary source. Count rising and falling edges of the primary source. Count rising edges of the primary source while the secondary input is high active. Quadrature count mode, uses primary clock and secondary input. Count rising edges of the primary clock; secondary input specifies direction (1 = minus). Edge of the secondary input triggers primary count until a compare occurs. Cascaded timer mode (up/down).

Table 21-4. TMR[0–3]CTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
PCS 12–9	0	**Primary Count Source** Select the primary count source. A timer selecting its own output for input is not a legal choice. The result is no counting.	0000 Timer 0 input signal. 0001 Timer 1 input signal. 0010 Timer 2 input signal. 0011 Timer 3 input signal. 0100 Timer 0 output for cascaded timer operation. 0101 Timer 1 output for cascaded timer operation. 0110 Timer 2 output for cascaded timer operation. 0111 Timer 3 output for cascaded timer operation. 1000 Prescaler ((CLASS clock/2)/1). 1001 Prescaler ((CLASS clock/2)/2). 1010 Prescaler ((CLASS clock/2)/4). 1011 Prescaler ((CLASS clock/2)/ 8). 1100 Prescaler ((CLASS clock/2)/16). 1101 Prescaler ((CLASS clock/2)/32). 1110 Prescaler ((CLASS clock/2)/64). 1111 Prescaler ((CLASS clock/2)/128).
SC 8–7	0	**Secondary Count Source** Provides additional information, such as direction, used for counting. These bits also define the source used by both the Capture mode bits and the input Edge Flag in the Timer Channel Status and Control register. The Timer n input signals are inputs of the timers in the quad timer module.	00 Timer 0 input signal. 01 Timer 1 input signal. 10 Timer 2 input signal. 11 Timer 3 input signal.
ONCE 6	0	**Count Once** Selects continuous or one-shot counting. If counting up, a successful compare occurs when the timer reaches TMRxCMP1 value. If counting down, a successful compare occurs when a timer reaches TMRxCMP2 value.	0 Count repeatedly. 1 Count to the compare value and then stop.
LEN 5	0	**Count Length** Determines whether the timer counts to the compare value and then reinitializes itself, or the timer continues counting past the compare value (binary roll-over). If counting up, a successful compare occurs when the timer reaches the TMRxCMP1 value. If counting down, a successful compare occurs when the timer reaches the TMRxCMP2 value.	0 Roll-over. 1 Count to the compare value and then reinitialize.
DIR 4	0	**Count Direction** Selects either the normal count up direction, or the reverse down direction.	0 Count up. 1 Count down.

Table 21-4. TMR[0–3]CTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
EIN 3	0	**External Initialization** Enables another timer within the same module to force the reinitialization of this timer when the other timer has an active compare event. Details on Broadcast mode are presented in **Section 21.1.5.3**, *Broadcast from an Initiator Timer,* on page 21-12.	0 External timers can not force a reinitialization of this timer. 1 External timers may force a reinitialization of this timer.
OFLM 2–0	0	**Output Mode** Determine the mode of operation for the timer output signal. For all of these modes except 000 and 111, the output flag is not toggled when the timer reaches the compare value but instead when the timer advances one value beyond. For example, for a compare value of 7, it toggles on the transition from 7 to 8, not 6 to 7. Unexpected results may occur if the Output mode field is set to use alternating compare registers (mode 100) and the ONCE bit is set.	000 Asserted while timer is active. 001 Clear timer output on successful compare. 010 Set timer output on a successful compare. 011 Toggle the timer output flag when a successful compare occurs. 100 Toggle the timer output flag using alternating compare registers. 101 Set on compare, cleared on secondary input signal's edge. 110 Set on compare, cleared on timer rollover. 111 Enable gated clock output while the timer is active.

21.4.1.2 Timer Channel Status and Control Registers (TMRnSCTLx)

TMR[0–3]SCTL[0–3] Timer Channel Status and Control Register Offset 0x1C + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TCF	TCFIE	TOF	TOFIE	IEF	IEFIE	IPS	INPUT	CM		MSTR	EEOF	VAL	FORC	OPS	OEN
Type	R/W							R	R/W					W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-5. TMR[0–3]SCTL[0–3] Bit Descriptions

Name	Reset	Description	Settings
TCF 15	0	**Timer Compare Flag** Set when a successful compare occurs. Clear the bit by writing 0 to it.	0 No successful compare. 1 Successful compare.
TCFIE 14	0	**Timer Compare Flag Interrupt Enable** Enables interrupts when the TCF bit is set.	0 No interrupt. 1 Interrupt.
TOF 13	0	**Timer Overflow Flag** Set when the timer rolls over its maximum value 0xFFFF or 0x0000, depending on count direction. Clear the bit by writing 0 to it.	0 No overflow. 1 Overflow. The timer has reached its maximum or minimum value.
TOFIE 12	0	**Timer Overflow Flag Interrupt Enable** Enables interrupts when the TOF bit is set.	0 No interrupt. 1 Interrupt.

Table 21-5. TMR[0–3]SCTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
IEF 11	0	**Input Edge Flag** Set when a positive input transition occurs while the timer is enabled. Clear the bit by writing a 0 to it. Setting the input polarity select (TxSCTL[IPS]) bit enables the detection of negative input edge transitions. Also, the control register secondary count source determines which external input pin is monitored by the detection circuitry.	0 No action. 1 Positive input transition while timer enabled.
IEFIE 10	0	**Input Edge Flag Interrupt Enable** Enables interrupts when the IEF bit is set.	0 No action. 1 Interrupts enabled.
IPS 9	0	**Input Polarity Select** Inverts the input signal polarity.	0 No action. 1 Invert signal polarity.
INPUT 8	0	**Secondary Input Signal** Reflects the current state of the secondary Input signal.	00 Capture function disabled. 01 Load capture register on rising edge of the secondary count source input. 10 Load capture register on falling edge of the secondary count source input. 11 Load capture register on any edge of the secondary count source input.
CM 7–6	0	**Input Capture Mode** Specifies the operation of the capture register as well as the operation of the input edge flag.	00 Capture function is disabled. 01 Load capture register on the rising edge of the secondary count source input. 10 Load capture register on the falling edge of the secondary count source input. 11 Load capture register on any edge of the secondary count source input.
MSTR 5	0	**Initiator Mode** Enables the compare function output to broadcast to the other timers in the module. This identifies a timer as the initiator timer in Broadcast mode. This signal is used to reinitialize the other timers and/or force their outputs. For details on Broadcast mode, see **Section 21.1.5.3**, *Broadcast from an Initiator Timer,* on page 21-12.	0 No action. 1 Broadcast mode.
EEOF 4	0	**Enable External Output Force** Enables the compare from another timer configured as the initiator to force the state of this timer output signal. For details on Broadcast mode, see **Section 21.1.5.3**, *Broadcast from an Initiator Timer,* on page 21-12.	o No action. 1 Other timer can force this timer's output flag signal.
VAL 3	0	**Forced Output Flag Value** Determines the value of the timer output flag signal when a software-triggered FORCE command occurs.	

Table 21-5. TMR[0–3]SCTL[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
FORC 2	0	**Force Output** Forces the current value of the VAL bit to be written to the timer output. Always read this bit as a 0. The VAL and FORCE bits can be written simultaneously in a single write operation. Write to the FORCE bit only if the timer is disabled. Setting this bit while the timer is enabled may yield unpredictable results.	0 No action. 1 Forces the current value of the VAL bit to be written to timer output.
OPS 1	0	**Output Polarity Select** Determines the polarity of the output signal.	0 True polarity. 1 Inverted polarity.
OEN 0	0	**Output Enable** Enables the timer output. The OPS bit determines the polarity of the output.	0 Timer output not enabled. 1 Timer output enabled.

21.4.1.3 Timer Channel Compare 1 Registers (TMRnCMP1x)

TMR[0–3]CMP1[0–3] Timer Channel Compare 1 Registers Offset x*0x40

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

TMR[0–3]CMP1[0–3] store the comparison value (CV) for comparison with the timer value.

21.4.1.4 Timer Channel Compare 2 Registers (TMRnCMP2x)

TMR[0–3]CMP2[0–3] Timer Channel Compare 2 Registers Offset 0x04 + x*0x40

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

TMR[0–3]CMP2[0–3] store the comparison value (CV) for comparison with the timer value.

21.4.1.5 Timer Channel Compare Load 1 Registers (TMRnCMPLD1x)

TMR[0–3]CMPLD1[0–3] Timer Channel Compare Load 1 Registers Offset 0x20 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLV1															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR[0–3]CMPLD1[0–3] store the preload value for the TMR[0–3]CMP1[0–3].

21.4.1.6 Timer Channel Compare Load 2 Registers (TMRnCMPLD2x)

TMR[0–3]CMPLD2[0–3] Timer Channel Compare Load 2 Registers Offset 0x24 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLV2															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR[0–3]CMPLD2[0–3] store the preload value for the TMR[0–3]CMP2[0–3].

21.4.1.7 Timer Channel Comparator Status and Control Registers (TMRnCOMSCx)

TMR[0–3]COMSC[0–3] Timer Channel Comparator Status and Control Registers Offset 0x28 + x*0x40

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—								TCF2EN	TCF1EN	TCF2	TCF1	CL2		CL1	
Type	R								R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TMR[0–3]COMSC[0–3] store the preload values used for the TMR[0–3]CMP2[0–3] registers.

Table 21-6. TMR[0–3]COMSC[0–3] Bit Descriptions

Name	Reset	Description	Settings
— 15–8	0	Reserved. Write to 0 for future compatibility.	
TCF2EN 7	0	**Timer Compare 2 Interrupt Enable** Enables the compare 2 interrupt. An interrupt is issued when both this bit and the TCF2 bit are set.	0 No action. 1 Enable compare 2 interrupt.

Table 21-6. TMR[0–3]COMSC[0–3] Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCF1EN 6	0	**Timer Compare 1 Interrupt Enable** Enables the compare 1 interrupt. An interrupt is issued when both this bit and the TCF1 bit are set.	0 No action. 1 Enable compare 1 interrupt.
TCF2 5	0	**Timer Compare 2 Interrupt Source** Indicates a successful comparison of the timer and the TMRxCMP2. This bit is sticky and remains set until it is explicitly cleared by writing a zero to this bit location.	0 Normal operation. 1 Successful compare 2.
TCF1 4	0	**Timer Compare 1 Interrupt Source** Indicates a successful comparison of the timer and the TMRxCMP1. This bit is sticky and remains set until it is explicitly cleared by writing a zero to this bit location.	0 Normal operation. 1 Successful compare 1.
CL2 3–2	0	**Compare Load Control 2** Control when TMRxCMP2 is preloaded with the value from TMRxCMPLD2.	00 Never preload. 01 Load upon successful compare with the value in TMRxCMP1. 10 Load upon successful compare with the value in TMRxCMP2. 11 Reserved.
CL1 1–0	0	**Compare Load Control 1** Control when TMRxCMP1 is preloaded with the value from TMRxCMPLD1.	00 Never preload. 01 Load upon successful compare with the value in TMRxCMP1. 10 Load upon successful compare with the value in TMRxCMP2. 11 Reserved.

21.4.1.8 Timer Channel Capture Registers (TMRnCAPx)

TMR[0–3]CAP[0–3] Timer Channel Capture Registers Offset 0x08 + x*0x40

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

TMR[0–3]CAP[0–3] store the values captured from the timers.

21.4.1.9 Timer Channel Load Registers (TMRnLOADx)

TMR[0–3]LOAD[0–3] Timer Channel Load Registers Offset 0x0C + x*0x40

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

TMR[0–3]LOAD[0–3] store the value used to load the timer.

21.4.1.10 Timer Channel Hold Registers (TMRnHOLDx)

TMR[0–3]HOLD[0–3] Timer Channel Hold Registers Offset 0x10 + x*0x40

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

TMR[0–3]HOLD[0–3] store the value whenever any timer is read.

21.4.1.11 Timer Channel Counter Registers (TMRnCNTRx)

TMR[0–3]CNTR[0–3] Timer Channel Counter Registers Offset 0x14 + x*0x40

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

TMR[0–3]CNTR[0–3] are counters.

21.4.2 SC3850 DSP Core Subsystem Timers

For a detailed information about programming the core subsystem timers, see the *MSC8156 SC3850 DSP Core Subsystem Reference Manual.*

21.4.3 Software Watchdog Timers

This section describes the software WDT registers, which include:

- System Watchdog Control Register 0–7 (SWCRR[0–7]), see page 21-26.
- System Watchdog Count Register 0–7 (SWCNR[0–7]), see page 21-27.
- System Watchdog Service Register 0–7 (SWSRR[0–7]), see page 21-27.

Note: The watchdog timers use the following base addresses:

- System Watchdog Timer 0 = 0xFFF25000
- System Watchdog Timer 1 = 0xFFF25100
- System Watchdog Timer 2 = 0xFFF25200
- System Watchdog Timer 3 = 0xFFF25300
- System Watchdog Timer 4 = 0xFFF25400
- System Watchdog Timer 5 = 0xFFF25500
- System Watchdog Timer 6 = 0xFFF25600
- System Watchdog Timer 7 = 0xFFF25700

21.4.3.1 System Watchdog Control Register 0–7 (SWCRR[0–7])

SWCRR[0–7] System Watchdog Control Register 0–7 Offset 0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SWTC															
Type	R/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
	SWTC															
	—													SWEN	SWRI	SWPR
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWCRR[0–7] control the software watchdog timer period and configure WDT operation. SWCRR can be read at any time but can be written only once after system reset. **Table 21-7** defines the SWCRR[0–7] bit fields.

Table 21-7. SWCRR[0–7] Bit Descriptions

Name	Reset	Description	Settings
SWTC 31–16	0xFFFF	**Software Watchdog Time Count** The SWTC field contains the modulus that is reloaded into the watchdog counter by a service sequence. When a new value is loaded into SWCRR[SWTC], the software watchdog timer is not updated until the servicing sequence is written to the SWSRR. If SWCRR[SWEN] is loaded with 0, the modulus counter does not count. The new value is also used at the next and all subsequent reloads. Reading the SWCRR register returns the value in the System Watchdog Control Register. Reset initializes the SWCRR[SWTC] field to $FFFF. **Note:** The prescaler counter is reset anytime a new value is loaded into the watchdog counter and also during reset.	
— 15–3	0	Reserved. Write to zero for future compatibility.	
SWEN 2	0	**Watchdog Enable** SWCRR[SWEN] bit enables the watchdog timer. It should be cleared by software after a system reset to disable the software watchdog timer. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state.	0 Watchdog timer disabled. 1 Watchdog timer enabled.
SWRI 1	0	**Software Watchdog Reset/Interrupt Select** Depending on the SWCRR[SWRI] programming, WDT timer causes a hard reset or machine check interrupt to the core.	0 Software watchdog timer causes a machine check interrupt to the core. 1 Software watchdog timer causes a hard reset (this is the default value after soft reset).
SWPR 0	0	**Software Watchdog Counter Prescale** Controls the divide-by-65536 WDT counter prescaler.	0 The WDT counter is not prescaled. 1 The WDT counter clock is prescaled.

21.4.3.2 System Watchdog Count Register 0–7 (SWCNR[0–7])

SWCNR[0–7] System Watchdog Count Register 0–7 Offset 0x08

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

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

SWCNR[0–7] provide visibility to the WDT counter values. Writes to SWCNR have no effect and terminate without transfer error exception.

Table 21-8. SWCNR[0–7] Bit Descriptions

Name	Reset	Description
— 31–16	0	Reserved. Write to zero for future compatibility.
SWCN 15–0	0xFFFF	**Software Watchdog Count Field** The read-only SWCNR[SWCN] field reflects the current value in the watchdog counter. Writing to the SWCNR register has no effect, and write cycles are terminated normally. Reset initializes the SWCNR[SWCN] field to $FFFF. Reading the 16 LS bits of 32-bit SWCNR register with two 8-bit reads is not guaranteed to return a coherent value.

21.4.3.3 System Watchdog Service Register 0–7 (SWSRR[0–7])

SWSRR[0–7] System Watchdog Service Register 0–7 Offset 0x0E

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

When the WDT is enabled, writing 0x556C followed by 0xAA39 to the SWSRR register before the WDT times out prevents a reset. If SWSRR is not serviced before the time-out, the watchdog timer sends a signal to the reset or interrupt controller module. Both writes must occur before the time-out in the order listed, but any number of instructions can be executed between the two writes. Reset initializes the SWSRR[WS] field to 0x0000. SWSRR can be written at any time, but returns all zeros when read. **Table 21-9** defines the bit fields of SWSRR.

Table 21-9. SWSRR[0–7] Bit Descriptions

Name	Reset	Description
WS 15–0	0	**Software Watchdog Service Field** Write 0x556C followed by 0xAA39 to this register to prevent software watchdog timer time-out. SWSRR[WS] can be written at any time, but returns all zeros when read. Writing any other value resets the servicing sequence.

GPIO 22

The MSC8156E device has 32 general-purpose I/O (GPIO) ports, 17 of which are multiplexed as either GPIO ports or dedicated peripheral interface ports. In addition, sixteen of the GPIOs can be configured as external maskable interrupt inputs. As GPIOs, each port is configured as an input or output (with a register for data output that is read or written at any time). If configured as output, the GPIO ports can also be configured as open-drain (that is, configured in an active low wired-OR configuration on the board). In this mode, an output drives a zero voltage but goes to tri-state when driving a high voltage. GPIO ports have configurable internal pull-up resistors. These resistors are enabled by default. They can be disabled by programming the GPIO Pull-Up Enable Register (GPUER); see the **Chapter 8**, *General Configuration Registers* for details. The dedicated MSC8156E peripheral functions multiplexed with the GPIO ports are grouped to maximize the usefulness of the ports in the greatest number of MSC8156E applications.

Note: To understand the port assignment capability described in this chapter, you must also understand the operation of the SPI, timers, UART, I^2C, and DMA peripherals.

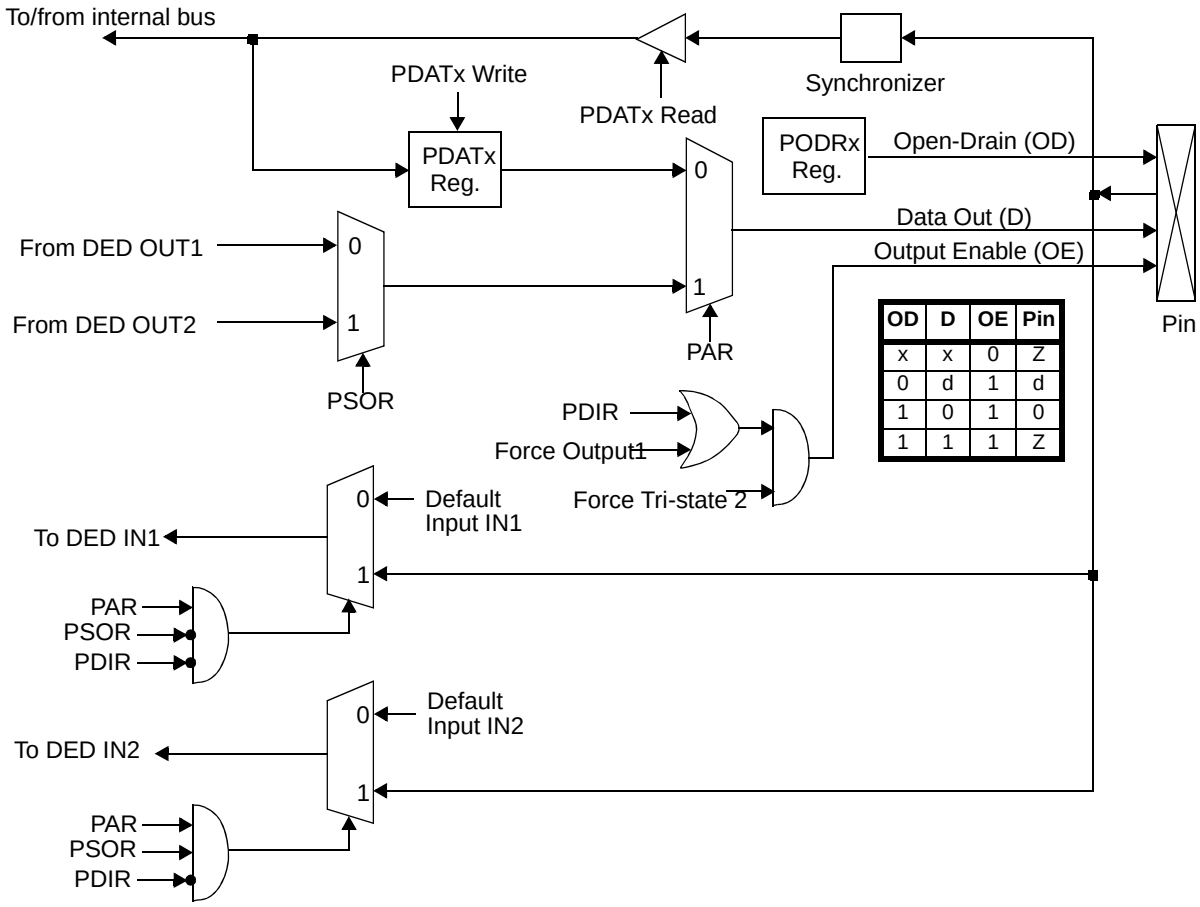
22.1 Features

Following are the key features of the GPIO ports:

- 32 GPIO ports.
- All ports are bidirectional.
- Seventeen ports have alternate on-device peripheral functions.
- At system reset, all 32 port inputs and outputs are disabled and the pull-up resistors are enabled.
- All port values can be read while the port is connected to an internal peripheral.
- All ports have open-drain output capability.
- Sixteen of the GPIOs can function as interrupt inputs.
- Four of the GPIOs can function as SPI signals.
- Five of the GPIOs can function as timer signals.
- Two of the GPIOs can function as a UART port.
- Two of the GPIOs can function as an I^2C interface.
- Four of the GPIOs can function as external DMA control signals.

22.2 GPIO Block Diagram

Figure 22-1 shows a GPIO functional block diagram.



Notes: 1. Force Output may be asserted high by dedicated peripheral 2 direction control, only when PAR = 1 and PSOR = 1 (selects this peripheral) and PDIR = 0 (input). It is used for bidirectional operation allowing the peripheral to dynamically control the port direction.
2. Force Tri-state may be asserted low by dedicated peripheral 1 output enable, only when PAR = 1 and PSOR = 0 (selects this peripheral) and PDIR = 1 (output). It is used for tri-state output operation, allowing the peripheral to control the port drive dynamically.

Port Control Register Bits

Register Name	0	1	Description
PARx	General-purpose	Dedicated	Port Assignment Registers
PSORx	Dedicated 1	Dedicated 2	Port Special Options Registers
PDIRx	Input	Output	Port Data Direction Registers
PODRx	Regular	Open drain	Port Open-Drain Registers
PDATx	0 (data)	1 (data)	Port Data Registers
Note: In addition to these registers, you must set each the ports as inputs using the General Input Enable (GIER) register. See **8.2.10**, *GPIO Input Enable Register (GIER)* for details.			

Figure 22-1. Port Functional Operation

22.3 GPIO Connection Functions

This section describes the GPIO port when it has GPIO or dedicated functionality, which depends on the settings in the Pin Assignment Register (PAR), as follows:

- Each port is independently configured as a GPIO if the corresponding PAR bit is cleared. A port is configured as an input if the corresponding control bit in the Pin Data Direction Register (PDIR) is cleared; it is configured as an output if the corresponding PDIR bit is set.
- Each port is configured as a dedicated on-device peripheral port if the corresponding PAR bit is set.

All PAR and PDIR bits are cleared on total system reset, which disables the output direction of the GPIO ports.

Data transfer is done through the Pin Data Register (PDAT). Data written to the PDAT is stored in an output register. If a GPIO is configured as an output, the output register data is gated onto the GPIO port. If a GPIO is configured as an input and the port is enabled by the appropriate bit in the GIER, a read of PDAT*x* is actually a read of the GPIO port itself. Data written to PDAT when the GPIO is configured as an input is still stored in the output register, but it is prevented from reaching the external port.

When a multiplexed GPIO port is not configured as a GPIO, it has a dedicated functionality, as described in **Table 22-1**. If an input to a peripheral is not supplied externally, a default value is supplied to the internal peripheral as listed in the right-most column.

Table 22-1 describes the functionality of the GPIO ports according to the configuration of the port registers (PAR, PSOR, and PDIR). Each port can be configured as a GPIO (input, regular output, or open-drain output), one of two dedicated outputs, or one of two dedicated inputs. A route of one GPIO-dedicated output to another GPIO-dedicated input gives even more flexibility. The implemented routing is described in **Table 22-1** as primary and secondary input.

Table 22-1. GPIO Dedicated Assignment (PARx = 1)

GPIO	Port Function					
	PSORx = 0			PSORx = 1		
	PDIRx = 1 (Output, Unless Tri-State Option is Specified)	PDIRx = 0 (Input)	Default Input	PDIRx = 1 (Output)	PDIRx = 0 (Input, Unless Inout Option is Specified)	Default Input
0	—	$\overline{\text{IRQ0}}$	1	—	—	0
1	—	$\overline{\text{IRQ1}}$	1	—	—	0
2	—	$\overline{\text{IRQ2}}$	1	—	—	0
3	—	$\overline{\text{IRQ3}}$	1	—	DRQ1	0
4	—	$\overline{\text{IRQ4}}$	1	DDN1	—	0

Table 22-1. GPIO Dedicated Assignment (PARx = 1) (Continued)

GPIO	Port Function					
	PSORx = 0			PSORx = 1		
	PDIRx = 1 (Output, Unless Tri-State Option is Specified)	PDIRx = 0 (Input)	Default Input	PDIRx = 1 (Output)	PDIRx = 0 (Input, Unless Inout Option is Specified)	Default Input
5	—	$\overline{\text{IRQ5}}$	1	—	—	0
6	—	$\overline{\text{IRQ6}}$	1	—	—	0
7	—	$\overline{\text{IRQ7}}$	1	—	—	0
8	—	$\overline{\text{IRQ8}}$	1	—	—	0
9	—	$\overline{\text{IRQ9}}$	1	—	—	0
10	—	$\overline{\text{IRQ10}}$	1	—	—	0
11	—	$\overline{\text{IRQ11}}$	1	—	—	0
12	—	$\overline{\text{IRQ12}}$	1	—	—	0
13	—	$\overline{\text{IRQ13}}$	1	—	—	0
14	—	$\overline{\text{IRQ14}}$	1	—	DRQ0	0
15	—	$\overline{\text{IRQ15}}$	1	DDN0	—	0
16	—	—	0	—	—	0
17	—	—	0	SPI_SCK	SPI_SCK	0
18	—	—	0	SPI_MOSI	SPI_MOSI	0
19	—	—	0	SPI_MISO	SPI_MISO	0
20	—	—	0	$\overline{\text{SPI\_SL}}$	$\overline{\text{SPI\_SL}}$	1
21	—	—	0	—	—	0
22	—	—	0	—	—	0
23	—	—	0	TMR0	TMR0	0
24	—	—	0	TMR1	TMR1	0
25	—	—	0	TMR2	TMR2	0
26	—	—	0	TMR3	TMR3	0
27	—	—	0	TMR4	TMR4	0
28	—	—	0	UART_RXD	UART_RXD	1
29	—	—	0	UART_TXD	UART_TXD	0
30	—	—	0	I2C_SCL	I2C_SCL	0
31	—	—	0	I2C_SDA	I2C_SDA	0

Note: You must enable the port to use it as an input. See **8.2.10**, *GPIO Input Enable Register (GIER)* for details.

22.4 GPIO Programming Model

The GPIO registers reside on a 256 KB address space. The GPIO block has five memory-mapped, read/write, 32-bit control registers. This section describes these registers in detail. Following is a list of the GPIO registers:

- Pin Open-Drain Register (PODR), **page 22-6**
- Pin Data Register (PDAT), **page 22-7**
- Pin Data Direction Registers (PDIR), **page 22-8**
- Pin Assignment Register (PAR), **page 22-9**
- Pin Special Options Registers (PSOR), **page 22-10**

Note: The GPIO registers use a base address of: 0xFFF27200.

If the corresponding PAR[DD*x*] bit is 1 (configured as a dedicated peripheral function port) before a PSOR or PDIR bit is programmed, an external connection may function for a short period as an unwanted dedicated function and cause unpredictable behavior. Thus, it is recommended that you program the GPIO in the following sequence: PSOR, PODR, PDIR, PAR.

In addition to the GPIO registers, there are two registers in the set of General Configuration Registers (GCRs) that control GPIO operation. These are as follows:

- GPIO Pull-Up Enable Register (GPUER), see **Section 8.2.9**, *GPIO Pull-Up Enable Register (GPUER)*, on page 8-17.
- GPIO Input Enable Register (GIER), see **Section 8.2.10**, *GPIO Input Enable Register (GIER)*, on page 8-18. To be used as an input line, a port must be enabled by this register.

Note: The base address for the general configuration registers is: 0xFFF28000.

22.4.1 Pin Open-Drain Register (PODR)

PODR Pin Open-Drain Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OD31	OD30	OD29	OD28	OD27	OD26	OD25	OD24	OD23	OD22	OD21	OD20	OD19	OD18	OD17	OD16
Type	R/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
	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PODR indicates a normal or active low open drain mode for wired-OR configuration of the outputs.

Table 22-2. PODR Bit Descriptions

Name	Reset	Description	Settings
OD[31–0] 31–0	0	**Open-Drain Configuration** Determines whether the corresponding port is actively driven as an output or is an open-drain driver. As an open-drain driver, the port is driven active-low. Otherwise, it is tri-stated (high impedance).	0 The I/O port is actively driven as an output. 1 The I/O port is an open-drain driver.

Note: The GPIO registers use a base address of: 0xFFF27200.

22.4.2 Pin Data Register (PDAT)

PDAT Pin Data Register Offset 0x08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
Type	R/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
	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A read of a PDAT register returns the data at the pin if the input line is enabled by the correct bit in the GIER, independently of whether the port is defined as an input or output. Thus, output conflicts at the pin can be detected by comparing the written data with the data on the pin. A write to the PDAT*x* is sampled in a register bit, and if the equivalent PDIR bit is configured as an output, the value sampled for that bit is driven onto its respective pin. PDAT can be read or written at any time.

If a pin is selected as GPIO, it is accessed through the PDAT. Data written to the PDAT register is stored in an output register. If a port is configured as an output, the output register data is gated onto the pin. When PDAT is read, the GPIO pin itself is read. If a GPIO port is configured as an input and enabled by the correct bit in the GIER, data written to the PDAT register is still stored in the output register, but it is prevented from reaching the actual pin. When the PDAT register is read and enabled as an input, the state of the actual pin is read.

Note: The GPIO registers use a base address of: 0xFFF27200.

22.4.3 Pin Data Direction Register (PDIR)

PDIR Pin Data Direction Register Offset 0x10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DR31	DR30	DR29	DR28	DR27	DR26	DR25	DR24	DR23	DR22	DR21	DR20	DR19	DR18	DR17	DR16
Type	R/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
	DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PDIR is cleared at system reset.

Table 22-3. PDIR Bit Descriptions

Name	Reset	Description	Settings
DR 31–0	0	**Direction** Indicates whether a port is an input or an output.	0 The corresponding port is an input. 1 The corresponding port is an output.
Note: This register sets the direction of the selected port, but you must enable the port using the General Input Enable Register (GIER) to use it. See **Section 8.2.10**, *GPIO Input Enable Register (GIER)*, on page 8-18 for details.			

Note: The GPIO registers use a base address of: 0xFFF27200.

22.4.4 Pin Assignment Register (PAR)

PAR Pin Assignment Register Offset 0x18

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DD31	DD30	DD29	DD28	DD27	DD26	DD25	DD24	DD23	DD22	DD21	DD20	DD19	DD18	DD17	DD16
Type	R/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
	DD15	DD14	DD13	DD12	DD11	DD10	DD9	DD8	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAR is cleared at system reset.

Table 22-4. PAR Bit Descriptions

Name	Reset	Description	Bit Settings
DD[31–0] 31–0	0	**Dedicated Enable** Indicates whether a pin is a GPIO or a dedicated peripheral port. As a dedicated peripheral function, the pin is used by the internal module. The internal peripheral function to which it is dedicated can be determined by other bits, such as those in the PSOR.	0 GPIO. The peripheral functions of the pin are not used. 1 Dedicated peripheral function.

Note: The GPIO registers use a base address of: 0xFFF27200.

22.4.5 Pin Special Options Register (PSOR)

PSOR Pin Special Options Register Offset 0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SO31	SO30	SO29	SO28	SO27	SO26	SO25	SO24	SO23	SO22	SO21	SO20	SO19	SO18	SO17	SO16
Type	R/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
	SO15	SO14	SO13	SO12	SO11	SO10	SO9	SO8	SO7	SO6	SO5	SO4	SO3	SO2	SO1	SO0
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PSOR bits are effective only if the corresponding PAR[DD*x*] bit is 1 (a dedicated peripheral function).

Table 22-5. PSOR Bit Descriptions

Name	Reset	Description	Settings
SO[31–0] 31–0	0	**Special-Option** Determines whether an external connection configured for a dedicated function (PAR[DD] = 1) uses option 1 or option 2. See **Table 22-1**.	0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

Note: The GPIO registers use a base address of: 0xFFF27200.

Hardware Semaphores 23

The MSC8156E hardware semaphores (HS) hold eight coded hardware semaphores. A coded hardware semaphore provides a simple way to achieve a “lock” operation via a single write access, eliminating the need for such sophisticated read-modify-write mechanisms as the reservation. Using the hardware semaphores, external hosts and on-device bus initiators can protect internal and external shared resources and ensure coherency in any sequence of operations on these resources. Each coded hardware semaphore is an eight-bit register with a selective write mechanism, as follows (see **Figure 23-1**):

- The semaphore is *free* if its value is zero.
- The semaphore is *locked* if its value is non-zero and its value is the *lock code*. Each processor/task that needs the lock capability of the semaphore must have a unique non-zero eight-bit code for locking the semaphore for correct protocol operation.
- A write of a non-zero value (lock code) is successful only if the current value of the semaphore is zero (free). This write is defined as a successful *lock* operation, and the written value is the *lock code*.
- A write of a non-zero value (lock code) is ignored if the current value of the semaphore is non-zero (locked). This write is defined as a failed *lock* operation, since the coded semaphore is considered locked with the non-zero code it holds.
- To maintain the protocol coherency, only the initiator that successfully locked the semaphore is allowed (and obligated) to free it. A write of zero is always successful and is defined as a *free* operation.

When a processor/task must lock an unlocked semaphore to its unique code, it writes its unique lock code to the semaphore register. It then reads back the semaphore value, and if it matches its lock code, the lock operation is successful. A value mismatch (a different non-zero code or zero) indicates to the initiator that the lock operation failed and must be retried. After a successful lock operation, the initiator can do any coherent operation associated with this semaphore and then it must free the semaphore (write zero).

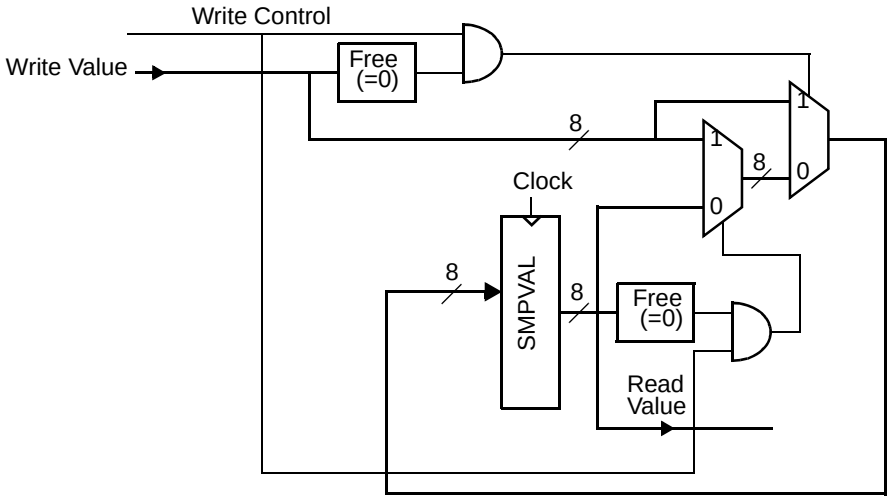


Figure 23-1. Hardware Semaphore Block Diagram

The hardware semaphore registers reside in the CCSR address space and have a constant offset. They are accessed through the MBus. The addresses of the hardware semaphore registers are presented in **Chapter 9**, *Memory Map*. The registers use a base address of 0xFFF27100.

HSMPR[0–7] Hardware Semaphore Register n Offset n*0x8

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

Table 23-1. HSMPRx Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
SMPVAL 7–0	0	**Semaphore Value** The eight-bit coded semaphore value. It holds the current semaphore value. A non-zero value indicates the current lock code.	0 Semaphore is free. It is writable to any value. ≠ 0 Semaphore is locked with lock code indicated by its current value. It is writable only to zero.

I²C 24

The inter-integrated circuit (IIC or I²C) bus is a two-wire—serial data (I2C_SDA) and serial clock (I2C_SCL)— bidirectional serial bus that provides a simple efficient method of data exchange between the system and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus minimizes the interconnections between devices. The synchronous, multi-initiator 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 initiators attempt to control the bus simultaneously. In the MSC8156E device, the maximum I2C_SCL frequency is 400 kHz. **Figure 24-1** is a block diagram of the I²C block.

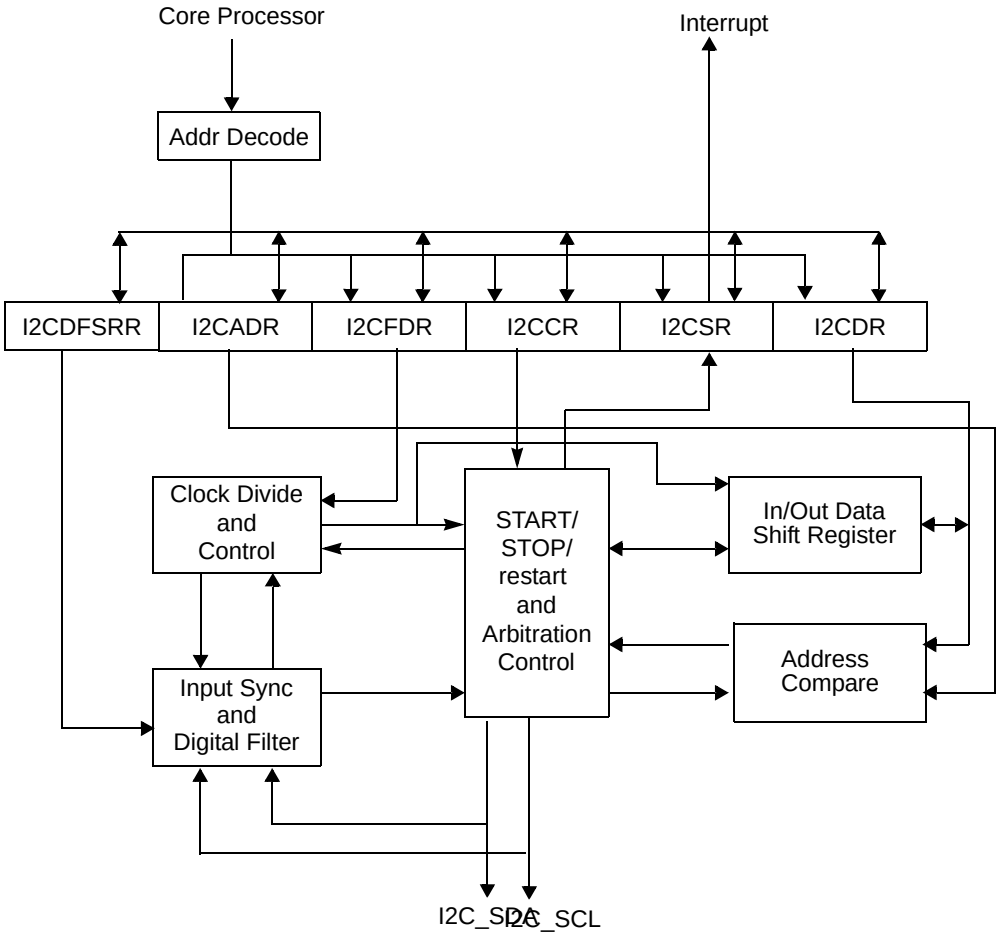


Figure 24-1. I²C Controller Block Diagram

24.1 Features

The I^2C interface includes the following features:

- Two-wire interface
- Multi-initiator operation
- Arbitration lost interrupt with automatic mode switching from initiator to target
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

24.2 Modes of Operation

The following modes of operation are supported by the I^2C controller:

- Initiator mode. The I^2C is the driver of the I2C_ line. It cannot use its own target address as a calling address. The I^2C cannot be an initiator and a target simultaneously. The initiator device initiates the data transfer by generating a START condition.
- The module must be enabled before a START condition from a I^2C initiator is detected.
- START condition. This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all targets. This mode is I^2C-specific.
- Repeated START condition. A START condition that is generated without a STOP condition to terminate the previous transfer. This mode is I^2C-specific.
- STOP condition. The initiator can terminate the transfer by generating a STOP condition to free the bus. This mode is I^2C-specific.
- Interrupt-driven byte-to-byte data transfer. When successful target addressing is achieved (and I2C_SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/W bit sent by the calling initiator. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode. Used only for loading the reset configuration word (RCW) only. See **Chapter 5**, *Reset*.

24.3 I²C Module Functional Blocks

The I²C module includes the following blocks:

- Clock control
- Input synchronization
- Digital input filter
- Transaction monitoring
- Arbitration control
- Transfer control
- In/out data shift register
- Address compare

24.3.1 Clock Control

The clock control block handles requests from clock for transferring and controlling data for multiple tasks. A 9-cycle data transfer (8-bit data plus the ACK bit) clock is requested for the following conditions:

- Initiator mode
  - transmit target address after START condition
  - transmit target address after restart condition
  - transmit data
  - receive data
- Target mode
  - transmit data
  - receive data
  - receive target address after START or restart condition

24.3.2 Input Synchronization

The input synchronization block synchronizes the input I2C_SCL and I2C_SDA signals to the [CLASS clock/2] and detects transitions of these signals.

24.3.3 Digital Input Filter

The I2C_SCL and I2C_SDA signal inputs are filtered to eliminate noise. Three consecutive signal samples are compared using a pre-determined sampling rate. If they are all high, the filter output is high. If they are all low, the output is low. For any high/low combination, the filter output is the value of the previous clock cycle. The sampling rate is the binary value stored in the frequency register. The sampling cycle duration is controlled by a down counter that sets a signal at the end of the count. This allows software to write to the frequency register to control the

filtered sampling rate. The value written to the I2CDFSRR should be defined by the system noise and the I2CFDR value. I2CDFSRR must be less than six times the division factor defined by I2CFDR. Note that the division factor stands for a I2CDFSRR value of 1.

24.3.4 Transaction Monitoring

The different conditions of the I^2C data transfers are monitored as follows:

- START conditions are detected when an I2C_SDA fall occurs while I2C_SCL is high.
- STOP conditions are detected when and I2C_SDA rise occurs while I2C_SCL is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a target address mismatch. Cancellation of data transactions resets the clock module
- The bus state is busy beginning with the detection of a START condition, and idle when a STOP condition is detected.

24.3.5 Arbitration Control

The arbitration control block controls the arbitration procedure of the initiator mode. A loss of arbitration occurs whenever the initiator detects a 0 on the external I2C_SDA line while attempting to drive a 1, tries to generate a START or restart at an inappropriate time, or detects an unexpected STOP request on the line. Arbitration by the initiator in initiator mode is lost under the following conditions:

- Low detected when high expected (transmit)
- Ack bit, low detected when high expected (receive)
- A START condition is attempted when the bus is busy
- A START condition is attempted when the bus is nearly busy (the I^2C controller does not yet recognize the bus as busy, but the bus is set to Initiator mode and I2C_SDA samples low).
- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

24.3.6 Transfer Control

The I^2C contains logic that controls the output to the serial data (I2C_SDA) and serial clock (I2C_SCL) lines of the I^2C. The I2C_SCL output is pulled low as determined by the internal clock generated in the clock module. The I2C_SDA output can only change at the midpoint of a low cycle of the I2C_SCL, unless performing a START, STOP, or restart condition. Otherwise, the I2C_SDA output is held constant. The I2C_SDA signal is pulled low when one or more of the following conditions are true in either initiator or target mode:

- Initiator mode
  - data bit (transmit)

- ack bit (receive)
- START condition
- STOP condition
- Restart condition

- Target mode
  - acknowledging address match
  - data bit (transmit)
  - ack bit (receive)

The I2C_SCL signal corresponds to the internal I2C_SCL signal when one or more of the following conditions are true in either initiator or target mode:

- Initiator mode
  - bus owner
  - lost arbitration
  - START condition
  - STOP condition
  - Restart condition begin
  - Restart condition end
- Target mode
  - address cycle
  - transmit cycle
  - ack cycle

24.3.7 In/Out Data Shift Register

This block controls the interface between the serial data line and the data register, both transmitting data to and receiving data from the I^2C. In transmit mode, a write to I2CCR[MTX] sets the direction to transmit. The contents of the parallel register are loaded into a shift register, which are then shifted on the I2C_SDA line.

24.3.8 Address Compare

The address compare block determines whether a target has been properly addressed and whether a specific action is required. The four performed address comparisons are described as follows:

- The address sent by the current initiator matches the general broadcast address (addresses all targets)
- The address matches the target address
- A broadcast message to update the I2CSR was received
- A message was addressed via the target address to update the I2CSR and to generate an interrupt

24.4 Functional Description

The reset state of the I^2C is the boot sequencer mode. After the boot sequencer has completed, the I^2C interface performs as a target receiver. The I^2C unit always performs as a target receiver as a default, unless explicitly programmed to be an initiator or target transmitter address.

A standard I^2C transfer consists of the following:

- START condition
- Target target address transmission (7-bit calling address plus the read/write bit)
- Data transfer
- STOP condition

Figure 24-2 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I^2C protocol. The details of the protocol are described in the following subsections.

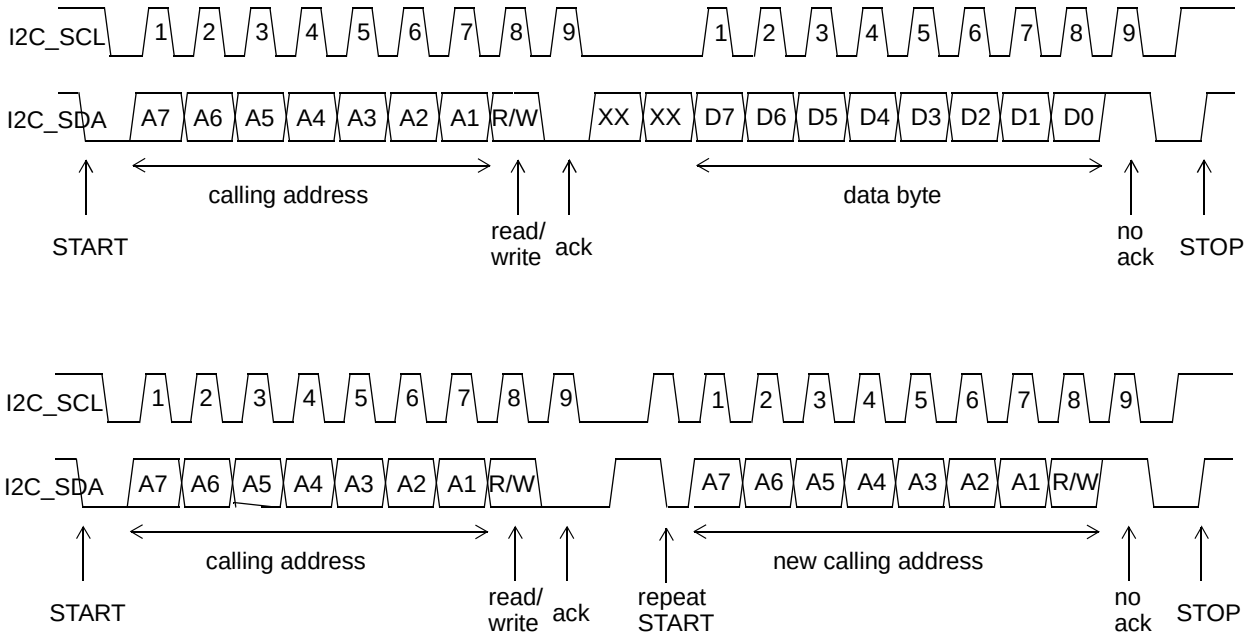


Figure 24-2. I^2C Interface Transaction Protocol

24.4.1 START Condition

When the I^2C bus is not engaged (both I2C_SDA and I2C_SCL lines are at logic high), an initiator can initiate a transfer by sending a START condition. As shown in **Figure 24-2**, a START condition is defined as a high-to-low transition of I2C_SDA while I2C_SCL is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all targets. The START condition is initiated by a software write that sets the I2CCR[MSTA].

24.4.2 Target Address Transmission

The first byte of data is transferred by the initiator immediately after the START condition is the target address. This is a seven-bit calling address followed by a R/W bit, which indicates the direction of the data being transferred to the target. Each target in the system has a unique address. In addition, when the I^2C module is operating as an initiator, it must not transmit an address that is the same as its target address. An I^2C device cannot be initiator and target at the same time. Only the target with a calling address that matches the one transmitted by the initiator responds by returning an acknowledge bit (pulling the I2C_SDA signal low at the 9th clock) as shown in **Figure 24-2**. If no target acknowledges the address, the initiator should generate a STOP condition or a repeated START condition. When target addressing is successful (and I2C_SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/W bit sent by the calling initiator.

The I^2C module responds to a general call (broadcast) command when I2CCR[BCST] is set. See the Philips I^2C specification for details. A broadcast address is always zero, however the I^2C module does not check the R/W bit. The second byte of the broadcast message is the initiator device ID. 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 device ID is for another receiver device and the third byte is a write command, then software can ignore the third byte during the broadcast. If the device ID 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. Each data byte is 8 bits long. Data bits can be changed only while I2C_SCL is low and must be held stable while I2C_SCL is high, as shown in **Figure 24-2**. There is one clock pulse on I2C_SCL for each data bit, and the most significant bit (msb) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device by pulling the I2C_SDA line low at the ninth clock. Therefore, one complete data byte transfer takes nine clock pulses. Several bytes can be transferred during a data transfer session. If the target receiver does not acknowledge the initiator, the I2C_SDA line must be left high by the target. The initiator can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling. If the initiator receiver does not acknowledge the target transmitter after a byte of transmission, the target interprets that the end-of-data has been reached. Then the target releases the I2C_SDA line for the initiator to generate a STOP or a START condition.

24.4.3 Repeated START Condition

Figure 24-2 shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The initiator uses this method to communicate with another target or with the same target in a different mode (transmit/receive mode) without releasing the bus.

24.4.4 STOP Condition

The initiator can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the I2C_SDA signal while I2C_SCL is high. For more information, see **Figure 24-2**. Note that an initiator can generate a STOP even if the target has transmitted an acknowledge bit, at which point the target must release the bus. The STOP condition is initiated by a software write that clears I2CCR[MSTA]. As described in **Section 24.4.3**, the initiator can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

24.4.5 Arbitration Procedure

The I^2C interface is a true multiple initiator bus that allows more than one initiator device to be connected on it. If two or more initiators simultaneously try to control the bus, each initiator's (including the I^2C module) clock synchronization procedure determines the bus clock—the low period is equal to the longest clock low period and the high is equal to the shortest one among the initiators. A bus initiator loses arbitration if it transmits a logic 1 on I2C_SDA while another initiator transmits a logic 0. The losing initiators immediately switch to target-receive mode and stop driving the I2C_SDA line. In this case, the transition from initiator to target mode does not generate a STOP condition. Meanwhile, the I^2C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a target, services the transaction if it is directed to itself.

Arbitration is lost (and I2CSR[MAL] is set) in the following circumstances:

- I2C_SDA sampled as low when the initiator drives a high during address or data-transmit cycle.
- I2C_SDA sampled as low when the initiator drives a high during the acknowledge bit of a data-receive cycle.
- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in target mode.

The I^2C module does not automatically retry a failed transfer attempt. If the I^2C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Target mode—the I^2C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected. If ICCR[MEN] is set and ICCR[RX] is cleared while the I2C_SDA signal is low and I2C_SCL is high, a false start condition is detected. If in this case, the data bits match the I^2C target address, then I2CSR[MAAS] is set. The application must correct the condition to release the I2C_SCL bus.

- Initiator mode—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 initiator of the I^2C interface losing arbitration, after which bus operations return to normal.

24.4.6 Clock Synchronization

Due to the wire AND logic on the I2C_SCL line, a high-to-low transition on the I2C_SCL line affects all devices connected on the bus. The devices begin counting their low period when the initiator drives the I2C_SCL line low. After a device has driven I2C_SCL low, it holds the I2C_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 I2C_SCL line if another device is still within its low period. Therefore, synchronized clock I2C_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 I2C_SCL line is released and pulled high. Then there is no difference between the device clocks and the state of the I2C_SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the I2C_SCL line low again.

24.4.7 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Target devices can hold the I2C_SCL low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the initiator clock into wait states until the target releases the I2C_SCL line.

24.4.8 Clock Stretching

Targets can use the clock synchronization mechanism to slow down the transfer bit rate. After the initiator has driven the I2C_SCL line low, the target can drive I2C_SCL low for the required period and then release it. If the target I2C_SCL low period is greater than the initiator I2C_SCL low period, then the resulting I2C_SCL bus signal low period is stretched.

24.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I^2C interface. It also includes **Figure 24-3**, a recommended flowchart for the I^2C interrupt service routines. The I^2C registers in this chapter are shown in big-endian format. If the system is in little-endian mode, software must swap the bytes appropriately. The I^2C controller does not guarantee its recovery from all illegal I^2C bus activity. In addition, 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.

24.5.1 Initialization Sequence

A hard reset initializes all the I^2C registers to their default states. The following initialization sequence initializes the I^2C unit:

- All I^2C registers must be located in a cache-inhibited page, that is the register locations must be defined as non-cacheable by the memory management units (MMUs).
- The GPIO registers must be configured to select the I^2C signal multiplexing options. See **Chapter 24**, *GPIO* for details.
- Update I2CFDR[FDR] and select the required division ratio to obtain the I2C_SCL frequency from the [CLASS clock/2] clock.
- Update I2CADR to define the target address for this device.
- Modify I2CCR to select initiator/target mode, transmit/receive mode, and interrupt-enable or disable.
- Set the I2CCR[MEN] to enable the I^2C interface.

24.5.2 Generation of START

After initialization, the following sequence can be used to generate START:

- If the device is connected to a multiinitiator I^2C system, test the state of I2CSR[MBB] to check whether the serial bus is free (I2CSR[MBB] = 0) before switching to initiator mode.
- Select initiator mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.
- Write the target address being called into the data register (I2CDR). The data written to I2CDR[7–1] comprises the target calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the target.

The scenario above assumes that the I^2C interrupt bit (I2CSR[MIF]) is cleared. If I2CSR[MIF] = 1 at any time, the I^2C interrupt handler should immediately handle the interrupt.

24.5.3 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. The I^2C interrupt bit (I2CSR[MIF]) is also set; an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] = 1). In the interrupt handler, software must do the following:

- Clear I2CSR[MIF]
- Read the contents of the I^2C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this causes I2CSR[MCF] to be cleared. See **Section 24.5.10** for details.

When an interrupt occurs at the end of the address cycle, the initiator remains in transmit mode. If initiator receive mode is required, I2CCR[MTX] should be toggled at this stage. See Section 24.5.10, “Interrupt Service Routine Flowchart.” If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] should 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). During target-mode address cycles (I2CSR[MAAS] = 1), I2CSR[SRW] should be read to determine the direction of the subsequent transfer and I2CCR[MTX] should be programmed accordingly. For target-mode data cycles (I2CSR[MAAS] = 0), I2CSR[SRW] is not valid and I2CCR[MTX] should be read to determine the direction of the current transfer. See **Section 24.5.10** for details.

24.5.4 Generation of STOP

A data transfer ends with a STOP condition generated by the initiator device. An initiator transmitter can generate a STOP condition after all the data has been transmitted. If an initiator receiver wants to terminate a data transfer, it must inform the target transmitter by not acknowledging the last byte of data, which is done by setting the transmit acknowledge (I2CCR[TXAK]) bit before reading the next-to-last byte of data At this time, the next-to-last byte of data has already been transferred on the I^2C interface, so the last byte will not receive the data acknowledge when I2CCR[TXAK] is set. For 1-byte transfers, a dummy read should be performed by the interrupt service routine. (See Section 24.5.10, “Interrupt Service Routine Flowchart.”) Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated. Eventually, I2CCR[TXAK] must be cleared again for subsequent I^2C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

24.5.5 Generation of Repeated START

At the end of a data transfer, if the initiator still wants to communicate on the bus, it can generate another START condition followed by another target address without first generating a STOP condition by setting I2CCR[RSTA].

24.5.6 Generation of I2C_SCL When I2C_SDA Low

In some cases it is necessary to force the I^2C module to become the I^2C bus initiator out of reset and drive the I2C_SCL signal (even though I2C_SDA may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I^2C devices to be reset. Thus, the I2C_SDA signal can be driven low by another I^2C device while the I^2C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force the I^2C module to generate I2C_SCL so that the device driving I2C_SDA can finish its transaction:

- Disable the I^2C and set the initiator bit by setting I2CCR to 0x20.
- Enable the I^2C by setting I2CCR to 0xA0.
- Read the I2CDR.
- Return the I^2C module to target mode by setting I2CCR to 0x80.

24.5.7 Target Mode Interrupt Service Routine

In the target interrupt service routine, the module addressed as a target should be tested to check if a calling of its own address has been received. If I2CSR[MAAS] = 1, software should set the transmit/receive mode select bit (I2CCR[MTX]) according to the R/$\overline{W}$ command bit (I2CSR[SRW]). Writing to I2CCR clears I2CSR[MAAS] automatically. The only time I2CSR[MAAS] is read as set is from the interrupt handler at the end of that address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have I2CSR[MAAS] = 0. A data transfer can then be initiated by writing to I2CDR for target transmits or dummy reading from I2CDR in target-receive mode. The target drives I2C_SCL low between byte transfers. I2C_SCL is released when the I2CDR is accessed in the required mode.

24.5.8 Target Transmitter and Received Acknowledge

In the target transmitter routine, the received acknowledge bit (I2CSR[RXAK]) must be tested before sending the next byte of data. The initiator signals an end-of-data by not acknowledging the data transfer from the target. When no acknowledge is received (I2CSR[RXAK] = 1), the target transmitter interrupt routine must clear I2CCR[MTX] to switch the target from transmitter to receiver mode. A dummy read of I2CDR then releases I2C_SCL so that the initiator can generate a STOP condition. See Section 24.5.10, “Interrupt Service Routine Flowchart.”

24.5.9 Loss of Arbitration and Forcing of Target Mode

When an initiator loses arbitration the following conditions all occur—

- I2CSR[MAL] is set
- I2CCR[MSTA] is cleared (changing the initiator to target mode)
- An interrupt occurs (if enabled) at the falling edge of the 9th clock of this transfer.

Thus, the target interrupt service routine should test I2CSR[MAL] first, and the software should clear I2CSR[MAL] if it is set. See **Section 24.3.5**, for details.

24.5.10 Interrupt Service Routine Flowchart

Figure 24-3 shows an example algorithm for an I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior. However, in the target receive mode (not shown in the flowchart), the interrupt service routine may need to set I2CCR[TXAK] when the

next-to-last byte is to be accepted. It is recommended that a sync instruction follow each I^{2}C register read or write to guarantee in-order instruction execution.

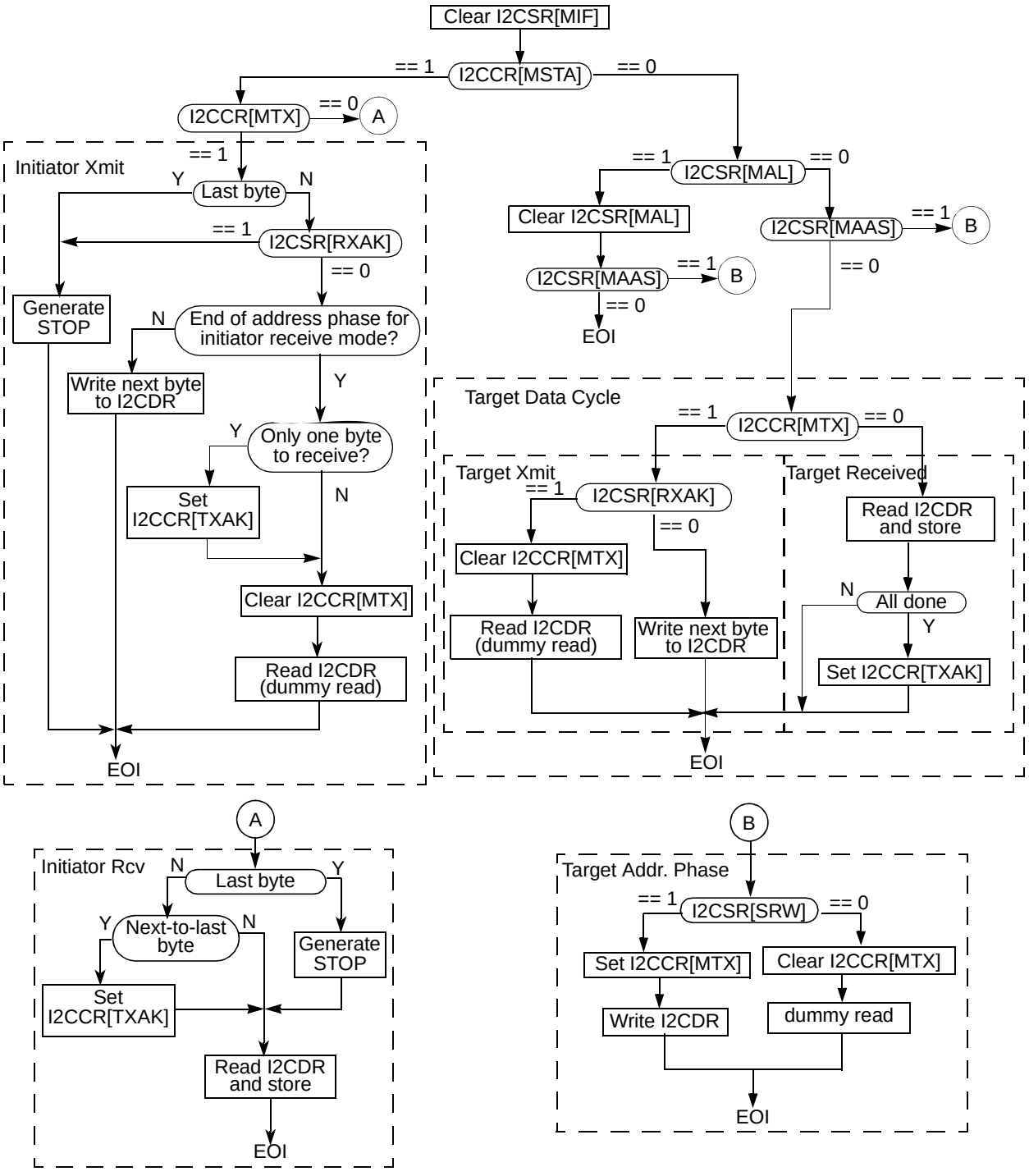


Figure 24-3. Example I^{2}C Interrupt Service Routine Flowchart

24.6 Programming Model

The registers covered in this section are as follows:

- I^{2}C Address Register (I2CADR), page 24-14
- I^{2}C Frequency Divider Register (I2CFDR), page 24-15
- I^{2}C Control Register (I2CCR), page 24-16
- I^{2}C Status Register (I2CSR), page 24-17
- I^{2}C Data Register (I2CDR), page 24-19
- Digital Filter Sampling Rate Register (I2CDFSRR), page 24-19

Note: Reserved bits should always be written with the value they returned when read. In other words, the register should be programmed by reading the value, modifying the appropriate fields, and writing back the value. The return value of the reserved fields should not be assumed, even though the reserved fields return zero. This note does not apply to the I^{2}C data register (I2CDR).

Note: The I^{2}C registers use a base address of: 0xFFF24C00. Accesses to the I^{2}C registers should be 1 byte in size and to addresses using the specified register offset.

24.6.1 I^{2}C Address Register (I2CADR)

I2CADR I^{2}C Address Register Offset 0x00

Bit	7	6	5	4	3	2	1	0
	ADDR							—
Type	R/W							
Reset	0	0	0	0	0	0	0	0

I2CADR contains the address to which the I^{2}C interface responds when addressed as a target. This is not the address sent on the bus during the address-calling cycle when the I^{2}C module is in initiator mode. **Table 24-1** describes the I2CADR fields.

Table 24-1. I2CADR Bit Descriptions

Name	Reset	Description
ADDR 7–1	0	**Target Address** Contains the specific target address used by the I^{2}C interface. The default mode of the I^{2}C interface is target mode for an address match.
— 0	0	Reserved. Write to zero for future compatibility.

24.6.2 I²C Frequency Divider Register (I2CFDR)

I2CFDR I2C Frequency Divider Register Offset 0x04

Bit	7	6	5	4	3	2	1	0
	—		FDR					
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Table 24-2 describes the I2CFDR fields. It also maps the I2CFDR[FDR] field to the clock divider values.

Table 24-2. I2CFDR Bit Descriptions

Name	Reset	Description	Settings			
— 7–6	0	Reserved. Write to zero for future compatibility.				
FDR 5–0	0	**Frequency Divider Ratio** Used to prescale the clock for bit rate selection. The serial bit clock frequency of I2C_SCL is equal to the [CLASS clock/2] clock divided by the divider. The frequency divider value can be changed at any point in a program.	**FDR**	**Divider (Decimal)**	**FDR**	**Divider (Decimal)**
			0x00	—	0x20	—
			0x01	—	0x21	—
			0x02	—	0x22	—
			0x03	—	0x23	—
			0x04	44	0x24	—
			0x05	52	0x25	—
			0x06	60	0x26	32
			0x07	64	0x27	36
			0x08	72	0x28	32
			0x09	88	0x29	40
			0x0A	104	0x2A	48
			0x0B	112	0x2B	56
			0x0C	128	0x2C	48
			0x0D	160	0x2D	64
			0x0E	192	0x2E	80
			0x0F	208	0x2F	96
			0x10	224	0x30	64
			0x11	288	0x31	96
			0x12	352	0x32	128
			0x13	384	0x33	160
			0x14	448	0x34	128
			0x15	576	0x35	192
			0x16	704	0x36	256
			0x17	768	0x37	320
			0x18	896	0x38	256
			0x19	1152	0x39	384
			0x1A	1408	0x3A	512
			0x1B	1536	0x3B	640
			0x1C	1792	0x3C	512
			0x1D	2304	0x3D	768
			0x1E	2816	0x3E	1024
			0x1F	3072	0x3F	1280

24.6.3 I²C Control Register (I2CCR)

I2CCR I²C Control Register Offset 0x08

Bit	7	6	5	4	3	2	1	0
	MEN	MIEN	MSTA	MTX	TXAK	RSTA	—	BCST
Type	R/W	R/W	W	R/W	R/W	W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Table 24-3 describes the I2CCR fields.

Table 24-3. I2CCR Bit Descriptions

Name	Reset	Description	Settings
MEN 7	0	**Module Enable** This bit controls the software reset of the I²C module. When cleared, the interface is held in reset but the registers can still be accessed. This bit must be set before any other control register bits have any effect. All I²C registers for target receive or initiator START can be initialized before setting this bit.	0 The module is reset and disabled. 1 The I²C module is enabled.
MIEN 6	0	**Module Interrupt Enable** Enables/disables interrupts from the I²C module. Clearing the bit does not clear any pending interrupts. When set, the interrupt occurs only if I2CSR[MIF] is also set.	0 Interrupts are disabled. 1 Interrupts are enabled.
MSTA 5	0	**Initiator/Target Mode START** Issues a STOP and changes to Target mode or a START and changes to Initiator mode. It the initiator loses arbitration, the bit is cleared without generating a STOP condition on the bus.	0 Issues a STOP condition and changes mode from initiator to target. 1 Issues a START condition and initiator mode is selected.
MTX 4	0	**Transmit/Receive Mode Select** This bit selects the direction of the initiator and target transfers. When configured as a target, this bit should be set by software according to I2CSR[SRW]. In initiator mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. The MTX bit is cleared when the initiator loses arbitration.	0 Receive mode. 1 Transmit mode.
TXAK 3	0	**Transfer Acknowledge** This bit specifies the value driven onto the I2C_SDA line during acknowledge cycles for both initiator and target receivers. The value of this bit only applies when the I²C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the core is addressed as a target, an acknowledge is always sent.	0 An acknowledge signal (I2C_SDA low) is sent to the bus on the 9th clock after receiving one byte of data. 1 No acknowledge signal response is sent (I2C_SDA high).
RSTA 2	0	**Repeat START** Setting this bit always generates a repeated START condition on the bus and provides the core with the current bus initiator. Attempting a repeated START at the wrong time (or if the bus is owned by another initiator), results in loss of arbitration.	0 No START condition generated. 1 Generates repeat START condition.

Table 24-3. I2CCR Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 1	0	Reserved. Write to zero for future compatibility.	
BCST 0	0	**Broadcast** Enables/disables the I^{2}C module to accept broadcast messages at address zero.	0 Broadcast disabled. 1 Broadcast enabled.

24.6.4 I^{2}C Status Register (I2CSR)

I2CSR I^{2}C Status Register Offset 0x0C

Bit	7	6	5	4	3	2	1	0
	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
Type	R	R	R	R/W	R	R	R/W	R
Reset	1	0	0	0	0	0	0	1

Table 24-4 describes the I2CSR fields.

Table 24-4. I2CSR Bit Descriptions

Name	Reset	Description	Settings
MCF 7	1	**Module Data Transfer Complete** When one byte of data is being transferred, the bit is cleared. It is set by the falling edge of the 9th clock of the byte transfer.	0 Byte transfer in progress. MCF is cleared under either of the following conditions: a. When I2CSR is read in receive mode or written in transmit mode, or b. after a START sequence is recognized by the I^{2}C controller in target mode. 1 Byte transfer is completed.
MAAS 6	0	**Module Address as Target** When the value in I2CDR matches the calling address, or the calling address matches the broadcast address (if broadcast mode is enabled), this bit is set. The processor is interrupted if I2CCR[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit.	0 Not addressed as target. 1 Addressed as target.
MBB 5	0	**Module 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 I^{2}C bus is idle. 1 I^{2}C bus is busy.
MAL 4	0	**Module Arbitration Lost** Automatically set when the arbitration procedure is lost. The core does not automatically retry a failed transfer attempt. This bit can only be cleared by software.	0 Arbitration is not lost. 1 Arbitration is lost.

Table 24-4. I2CSR Bit Descriptions (Continued)

Name	Reset	Description	Settings
BCSTM 3	0	**Broadcast Match** Writing to the I2CCR automatically clears this bit.	0 No broadcast match. 1 Calling address matches the broadcast address instead of the programmed target address, or the I^2C initiator drove an address of all 0s.
SRW 2	0	**Target Read/Write** When MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the initiator. This bit is valid only when both of the following conditions are true: • A complete transfer occurred and no other transfers have been initiated. • The I^2C interface is configured as a target and has an address match. By checking this bit, the processor can select target transmit/receive mode according to the command of the initiator.	0 Target receive, initiator writing to target. 1 Target transmit, initiator reading from target.
MIF 1	0	**Module Interrupt** The MIF bit is set when an interrupt is pending, causing a processor interrupt request if I2CCR[MIEN] is set. MIF is set when one of the following events occurs: • One byte of data is transferred (set at the falling edge of the 9th clock). • The value in I2CADR matches with the calling address in target-receive mode. • Arbitration is lost. The bit can only be cleared by software.	0 No interrupt pending. 1 Interrupt pending.
RXAK 0	1	**Received Acknowledge** The value of I2C_SDA during the reception of acknowledge bit of a bus cycle.	0 Acknowledge received after completion of 8-bit transmission on the bus. 1 No acknowledge detected on the ninth clock.

24.6.5 I²C Data Register (I2CDR)

I2CDR I²C Data Register Offset 0x10

Bit	7	6	5	4	3	2	1	0
	DATA							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Table 24-5 describes the I2CDR fields.

Table 24-5. I2CDR Bit Descriptions

Name	Reset	Description
DATA 7–0	0	**Data** Transmission starts when an address and the R/W bit are written to the data register and the I²C interface performs as the initiator. A data transfer is initiated when data is written to the I2CDR. The most significant bit is sent first in both cases. In the initiator receive mode, reading the data register allows the read to occur, but also allows the I²C module to receive the next byte of data on the I²C interface. In target mode, the same function is available after it is addressed.

24.6.6 Digital Filter Sampling Rate Register (I2CDFSRR)

I2CDFSRR Digital Filter Sampling Rate Register Offset 0x14

Bit	7	6	5	4	3	2	1	0
	—		DFSR					
Type	R/W							
Reset	0	0	0	1	0	0	0	0

Table 24-6 describes the I2CDFSRR fields.

Table 24-6. I2CDFSRR Bit Descriptions

Name	Reset	Description
— 7–6	0	Reserved. Write to zero for future compatibility.
DFSR 5–0	010000	**Digital Filter Sampling Rate** To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I²C bus. The resulting sampling rate is calculated by dividing the system frequency by the non-zero value of DFSR. If I2CDFSRR is set to zero, the I²C bus sample points default to the reset divisor 0x10. The value of DFSR should be defined by the system noise and the I2CFDR[FDR] value. DFSR must be less than six times the division factor defined by I2CFDR[FDR].

Debugging, Profiling, and Performance Monitoring 25

The MSC8156E device includes a number of mechanisms to evaluate and debug system operation. These features include:

- JTAG test access port (TAP) and boundary scan architecture
- On-chip emulator (OCE) module in each core
- Special debug and profiling elements in the device that permit:
  - Event counting
  - Entering debug mode after detection of a predefined state
  - Special trace modes in the QUICC Engine module and DSP core subsystems
  - Special debug errors
  - Host reads and writes of all registers in debug mode
- Performance monitoring of events occurring within internal modules including the DMA controller and RapidIO controller.

25.1 TAP, Boundary Scan, and OCE

The dedicated user-accessible test access port (TAP) is designed to be compatible with the **IEEE** Std. 1149.1 test access port and boundary scan architecture. Problems associated with testing high-density circuit boards led to development of this standard under the sponsorship of the test technology committee of **IEEE** and the joint test action group (JTAG). The MSC8156E supports circuit-board test strategies based on this standard. This section covers aspects of JTAG that are specific to the MSC8156E. It includes the items that the standard requires to be defined, with additional information specific to the MSC8156E device. For details on the standard, refer to the **IEEE** Std. 1149.1 documentation.

The JTAG port also provides access to the on-chip emulator (OCE) module, a dedicated block for debugging applications. Therefore, this section includes information on registers and functionality of the OCE module that are specific to the MSC8156E. For details on the OCE module functionality, see the *MSC8156 SC3850 DSP Core Subsystem Reference Manual*.

The SC3850 core OCE module interfaces with the SC3850 core and its peripherals non-intrusively so that you can examine registers, memory, or on-device peripherals, thus facilitating hardware and software development on the SC3850 core-based devices. Special

circuits and dedicated signals on the SC3850 core avoid sacrificing user-accessible internal resource. As the DSP applications grow in both size and complexity, the OCE module provides many features of the breakpoints, conditional breakpoints, breakpoints on data-bus values, and event detection that offer the user non-destructive access to peripherals, variety in profiling, a program tracing buffer, and real-time access to memory.

Note: There are some restrictions about using data breakpoints with some multi-variable move instructions. Multi-variable move instructions utilize the wide memory data bus to move several data elements in one access. For example, MOVE.2W transfers two 16-bit items as one 32-bit access. When the OCE data event detection channel (EDCD) is configured to detect a byte, word, or long reference data value and the core performs a multi-variable move, the reference value is searched in the positions on the bus where the variable could appear. In the MOVE.2W example, an EDCD search for a 16-bit value should be performed on bits 15–0 and 31–16 of the bus concurrently. However, for some SC3850 multi-variable byte move instructions, the EDCD does not check for the reference value in all optional positions of a byte. Therefore, it is possible that the EDCD byte breakpoints can be missed in these cases. The affected instructions include: MOVE2.2B, MOVE2.4B, MOVE2.8B, MOVEU2.2B, MOVEU2.4B, and MOVEU.8B.

25.1.1 Overview

The MSC8156E TAP consists of five dedicated signal lines, a 16-state TAP controller, and three test data registers. A Boundary Scan Register (BSR) links most of the device signal connections into a single shift register. The test logic, which uses static logic design, is independent of the device system logic. The MSC8156E JTAG can do the following:

- Perform boundary scan operations to check circuit-board electrical continuity.
- Bypass the MSC8156E for a given circuit-board test by effectively reducing the Boundary Scan Register (BSR) to a single cell.
- Sample the MSC8156E system connections during operation and transparently shift out the result in the BSR. Preload values to outputs prior to circuit board testing.
- Disable the drive to outputs during circuit board testing.
- Access the OCE controller and circuits to control a target system.
- Give entry to Debug mode.
- Query identification information (manufacturer, part number and version) from an MSC8156E-based device.
- Force test data onto the outputs of an MSC8156E-based device while replacing its BSR in the serial data path with a single-bit register.

Note: Precautions must be taken to ensure that the **IEEE** Std. 1149.1-like test logic does not interfere with non-test operation.

To access the JTAG registers, shift the appropriate command into the JTAG instruction register and then shift the required value into the register. See **Section 25.1.3** for a discussion of the JTAG instructions. **Figure 25-1** shows the MSC8156E JTAG 5-bit instruction register and the following test registers:

- Boundary Scan Register (BSR). Regarding the length of the BSR, The boundary scan bit definitions vary according to the specific chip implementation of the MSC8156E and are described by the BSDL file on the product website
- 1-bit Bypass Register
- 32-bit Identification Register (ID)
- 32 × 32-bit General-Purpose Register Bank (GSBI)
- Test port access register

Table 25-1 lists the test access port (TAP) signals.

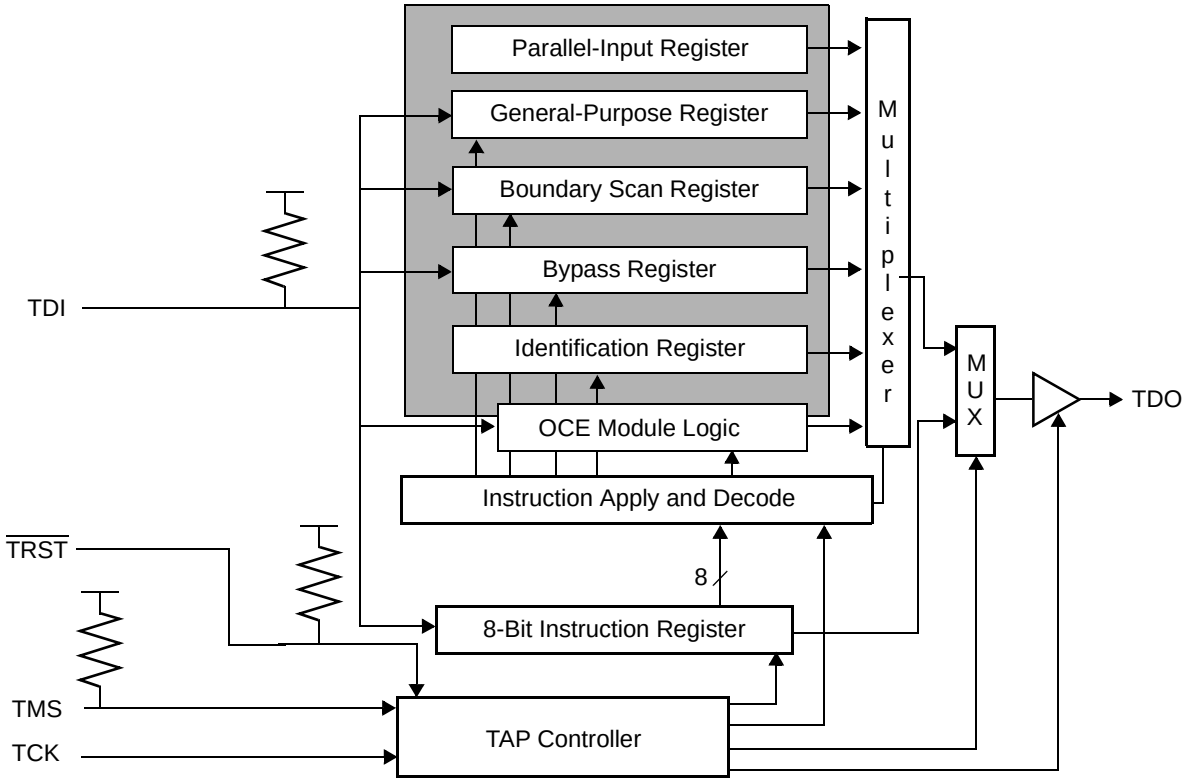


Figure 25-1. Test Logic Block Diagram

Table 25-1. TAP Signals

Signal	Description
TCK	A test clock input to synchronize the test logic.
TMS	A test mode select input (with an internal pull-up resistor) that is sampled on the rising edge of TCK to sequence the TAP controller state machine.
TDI	A test data input (with an internal pull-up resistor) that is sampled on the rising edge of TCK.

Table 25-1. TAP Signals (Continued)

Signal	Description
TDO	A data output that can be tri-stated and actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO changes on the falling edge of TCK.
$\overline{\text{TRST}}$	An asynchronous reset (with an internal pull-up resistor) that provides initialization of the TAP controller and other logic required by the standard.

25.1.2 TAP Controller

The TAP controller interprets the sequence of logical values on the TMS signal. This synchronous state machine controls the operation of the JTAG logic. The value adjacent to each arc in **Figure 25-2** represents the value of the TMS signal sampled on the rising edge of the TCK signal. For a description of the TAP controller states, refer to the **IEEE** Std. 1149.1 documentation.

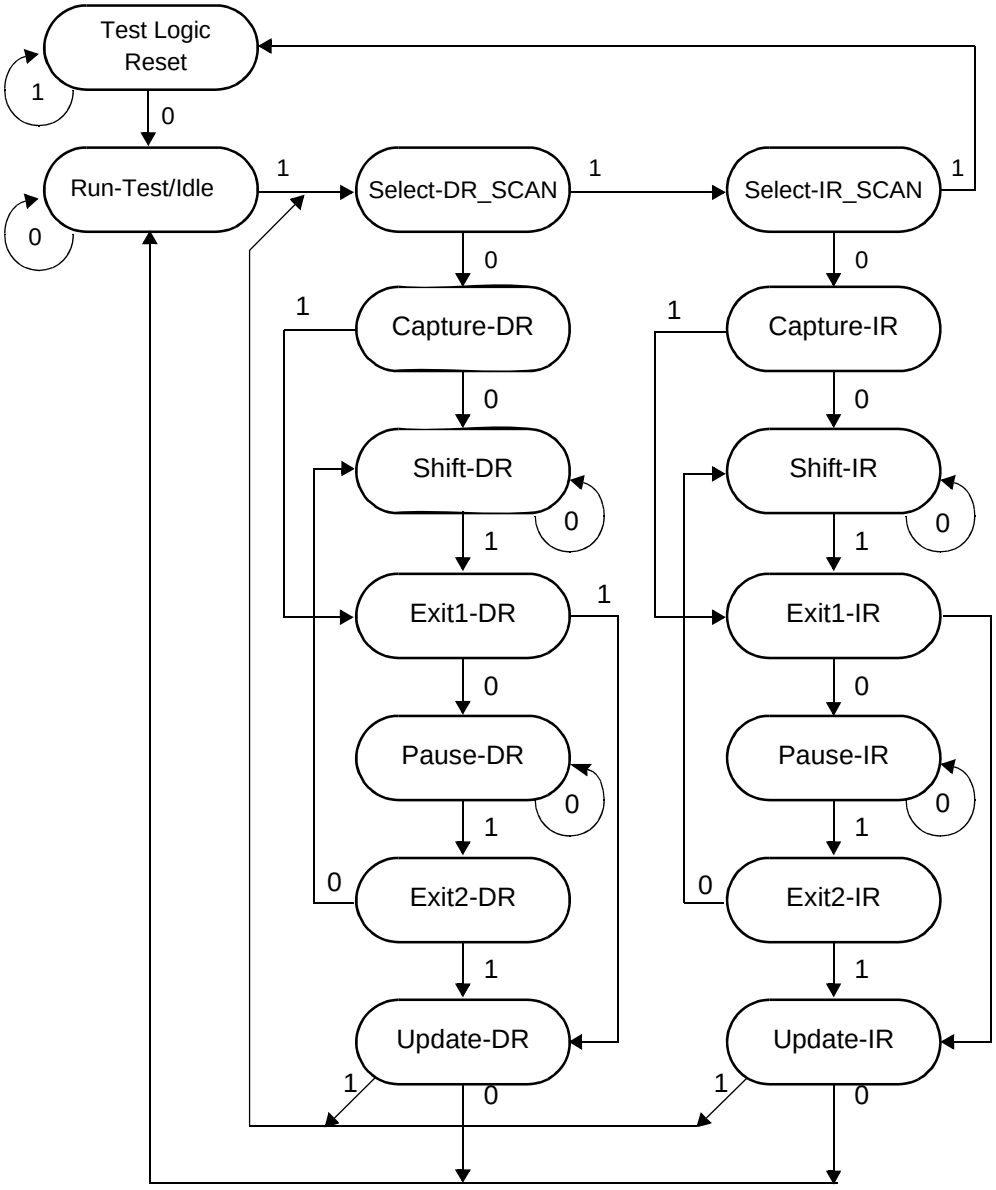


Figure 25-2. TAP Controller State Machine

25.1.3 Instruction Decoding

The MSC8156E includes the three mandatory public instructions EXTEST, SAMPLE/PRELOAD, and BYPASS and also supports the optional CLAMP and HIGHZ instructions defined by **IEEE Std.** 1149.1. The following public instructions perform key functions:

- READ_STATUS enables the JTAG port to query the status of the OCE circuitry.
- ENABLE_ONCE enables the JTAG port to communicate with the OCE circuitry.
- DEBUG_REQUEST enables the JTAG port to force the MSC8156E into Debug mode.
- CHOOSE_ONCE allows the operation of multiple OCE devices. This instruction should always execute before the first ENABLE_ONCE instruction and should shift a 1 to the SC3850 OCE module choose cells for each module that you want to enable. Since there are six internal OCE modules, you must shift 6 bits to the choose cells. For details, see **Section 25.1.4**.

The MSC8156E includes an 8-bit instruction register without parity, consisting of a shift register with eight parallel outputs. Data is transferred from the shift register to the parallel outputs during the UPDATE-IR controller state. The eight bits decode to the unique instructions listed in **Table 25-3**. All other encoding, with the exception of the manufacturer private instructions, is reserved for future enhancements and is decoded as BYPASS.

The parallel output of the Instruction Register is reset to 0xF3 in the test-logic-reset controller state, which is equivalent to the IDCODE instruction. During the CAPTURE-IR controller state, the parallel inputs to the instruction shift register are loaded with the code 01 in the least significant bits, as required by the standard. Two bits of the GPR are configured to select an SC3850 core, whose status is output from the multiplexer. Therefore, the status of all SC3850 cores can be viewed serially by updating the GPR between each SC3850 core status reading. Alternatively, all six SC3850 cores can be viewed simultaneously from the PIREG. For details on core states, refer to the *SC3850 StarCore DSP Reference Manual*.

Table 25-2. Instruction Register Capture and SC3850 Core Status Values

Name/bits	Description	Settings
FBiST_Done 7	**FBIST Done** Field built-in self test completed.	0 FBIST not complete. 1 FBIST completed
MBIST_Failed 6	**MBIST Failed** Indicates an MBIST failure.	0 No MBIST failure. 1 MBIST failed
MBIST_Done 5	**MBIST Done** Indicates that the MBIST is completed.	0 Either an activated MBIST is still running or no MBIST was initiated by the last $\overline{\text{HRESET}}$ 1 All active MBISTs are complete
— **4**	Reserved. Always 0.	
clock_end_count 3	**Clock End Count** Indicates that the counter in the clock block completed its count.	0 Clock block done signal not asserted 1 Clock block done signal asserted

Table 25-2. Instruction Register Capture and SC3850 Core Status Values

Name/bits	Description	Settings
retention-stopped 2	**Retention Stopped** Indicates whether all of the activated MBISTs in retention mode have stopped. This bit is cleared by assertion of HRESET or MBIST initiation.	0 An activated MBIST in retention mode has not stopped, or no MBIST retention stop was performed since the last HRESET assertion 1 All activated MBISTs in retention mode stopped.
— 1–0	Contains value (01) required by the JTAG standard	Read-only

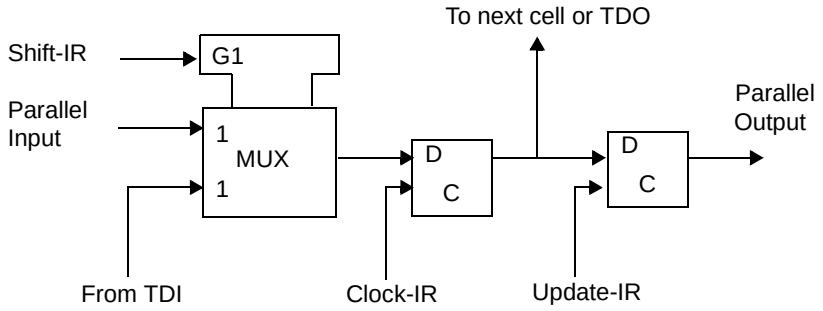


Figure 25-3. Instruction Register (IR) Configuration

Table 25-3 describes the 8-bit instructions coded in the Instruction Register.

Table 25-3. Instruction Decoding

Opcode	Instruction	Updates IR	Description
Standard Instructions			
0x00	EXTEST	Yes	Selects the Boundary Scan Register (BSR). EXTEST also asserts internal reset for the MSC8156E system logic to force a predictable internal state while external boundary scan operations are performed. By using the TAP, the register can: • Scan user-defined values into the output buffers • Capture values presented to inputs • Control the direction of bidirectional signals • Control the output drive of tri-statable outputs For details on the function and use of EXTEST, refer to the **IEEE** Std. 1149.1 documentation. Although the latest specification recommends not using all zeroes, it was mandated by earlier versions of the specification and is retained for backward compatibility.
0xF0	SAMPLE	Yes	SAMPLE provides a means to obtain a snapshot of system data and control signals.
0xF0	PRELOAD	Yes	Initializes the BSR output cells prior to the selection of EXTEST. This initialization ensures that known data appears on the outputs when an EXTEST instruction is entered.

Table 25-3. Instruction Decoding (Continued)

Opcode	Instruction	Updates IR	Description
0xF1	CLAMP	Yes	Optional in the **IEEE** Std. 1149.1. This public instruction selects the one-bit Bypass Register as the serial path between TDI and TDO, while allowing signals driven from the component to be determined from the Boundary Scan Register. During testing of ICs on PCBs, it may be necessary to place static guarding values on signals that control logic operations not involved in the test. The EXTEST instruction can be used for this purpose, but since it selects the BSR, the required guarding signals would be loaded as part of the complete serial data stream shifted in, both at the start of the test and each time a new test pattern is entered. Since the CLAMP instruction allows guarding values to be applied using the BSR of the appropriate ICs while selecting their Bypass Registers, it allows much faster testing than EXTEST. Data in the boundary scan cell remains unchanged until a new instruction is shifted in. **Note:** The CLAMP instruction also asserts internal reset for the MSC8156E system logic to force a predictable internal state while external boundary scan operations are performed.
0xF2	HIGHZ	Yes	Optional in the **IEEE** Std. 1149.1. It is a manufacturer's public instruction to prevent back-drive of the outputs during circuit-board testing. When HIGHZ is invoked, all output drivers, including the two-state drivers, are turned off (that is, high impedance). The HIGHZ instruction selects the Bypass Register. It also asserts internal reset for the MSC8156E system logic to force a predictable internal state while external boundary scan operations are performed.
0xF3	IDCODE	Yes	Selects the ID Register. This instruction is a public instruction to allow the manufacturer, part number and version of a component to be determined through the TAP. The ID Register configuration is as follows: • Bits 31–28: Version Information • Bits 27–12: Customer Part Number • Bits 11–1: Manufacturer Identity One application of the ID Register is to distinguish the manufacturer(s) of components on a board when multiple sourcing is used. included in the design. As required by the **IEEE** Std. 1149.1, the operation of the test logic has no effect on the operation of the internal system logic when the IDCODE instruction is selected. The value for the MSC8156E is 0x0189501D.
0xFE	STATUS	Yes	Private instruction that allows the user to scan out STATUS from the Instruction Register without changing the Instruction Update Register.
0xFF	BYPASS	Yes	Selects the single-bit Bypass Register. This creates a shift-register path from TDI to the Bypass Register and, finally, to TDO, circumventing the 573-bit BSR register. This instruction enhances test efficiency when a component other than the MSC8156E-based device is the device under test. When the current instruction selects the Bypass Register, the shift-register stage is set to a logic zero on the rising edge of TCK in the CAPTURE-DR controller state. Therefore, the first bit to be shifted out after the Bypass Register is selected is always a logic zero.
Clocking Control			
0x05	FREEZE	Yes	Private instruction that stops the clocks. The instructions causes all device clocks to stop. Resuming execution from this stopped state is not possible.

Table 25-3. Instruction Decoding (Continued)

Opcode	Instruction	Updates IR	Description
TLM			
0x03	TLM_SELECT	Yes	Private instruction that provides access to the TAP Linking Module.
0x04	TLM_HOLD	No	Private instruction that provides access to a TAP Linking Module. The most recent instruction that updated the Instruction Register is not overwritten by this instruction. Both the TLM Update Register and whatever else is selected by the instruction in the IR receive the TDI, but only the TLM Shift Register sends data to TDO.
OCE Instructions			
0xA0	ENABLE_ONCE	Yes	Not included in the **IEEE** Std. 1149.1. This public instruction allows you to perform system debug functions. When the ENABLE_ONCE instruction is decoded, TDI and TDO connect directly to the OCE registers. The OCE controller selects the specific OCE register connected between TDI and TDO, depending on the OCE instruction being executed. All communication with the OCE controller is through the SELECT-DR-SCAN path of the JTAG TAP Controller. Before the ENABLE_ONCE instruction is selected, the CHOOSE_ONCE instruction should be executed to define which OCE is to be activated. **Note:** This instruction is valid only if the core processor is running.
0xA1	DEBUG_REQUEST	Yes	Not included in the **IEEE** Std. 1149.1. This public instruction allows you to generate a debug request signal to the MSC8156E. When the DEBUG_REQUEST instruction is decoded, TDI and TDO connect to the OCE registers. In addition, ENABLE_ONCE is active and forced to request Debug mode from the MSC8156E to perform system debug functions. Before the DEBUG_REQUEST instruction is selected, the CHOOSE_ONCE instruction should be executed to define which OCE module is to be selected for DEBUG_REQUEST. **Note:** Issuing this instruction does not ensure that the SC3850 core enters the debug state. Monitor the core status to make sure that it has stopped.
0xA2	CHOOSE_ONCE	Yes	Not included in the **IEEE** Std. 1149.1. This instruction enables selected SC3850 OCE modules. All instructions executed after this one target only the selected OCE set. Therefore, this instruction always executes, regardless of the selected OCE set.
0xA3	RD_STATUS	Yes	The status of the OCE can be read from a dedicated status register inside the OCE by the JTAG instruction, RD_STATUS.

25.1.4 Multi-Core JTAG and OCE Module Concept

The MSC8156E uses JTAG TAP for standard defined testing compatibilities and for multi-core OCE module control and OCE module interconnection control. The MSC8156E has an internal OCE module per SC3850 core. The OCE modules interconnect in a chain and are configured and directed by the JTAG TAP controller. Each of the MSC8156E OCE modules has an interface to a JTAG port. The interface is active even when a reset signal to the SC3850 core is asserted. However, system reset must be deasserted to allow a proper interface with the cores. This interface is synchronized with the internal clocks derived from the JTAG TCK clock. Each OCE module includes an OCE controller, an event counter, an event detector unit, a synchronizer, an event selector, and a trace unit.

Note: For details on the OCE module features, consult the *MSC8156 SC3850 DSP Core Subsystem Reference Manual*.

The JTAG port performs the following tasks via the JTAG-OCE module interface:

- Chooses one or more OCE module blocks (CHOOSE_ONCE)
- Issues a debug request to the OCE module (DEBUG_REQUEST)
- Writes an OCE command to the OCE Command Register (DEBUG_REQUEST or ENABLE_ONCE)
- Reads and writes to internal OCE registers (DEBUG_REQUEST or ENABLE_ONCE)
- Queries the status of the OCE block (RD_STATUS)

25.1.5 Enabling the OCE Module

The CHOOSE_ONCE mechanism integrates multiple cores and thus multiple OCE modules on the same device. Using the CHOOSE_ONCE instruction, you can selectively activate one or more of the OCE modules on the MSC8156E. The OCE modules selected by the CHOOSE_ONCE instruction are cascaded as shown in **Figure 25-4**. Only selected OCE modules

respond to ENABLE_ONCE and DEBUG_REQUEST instructions from the JTAG. All OCE modules are deselected after reset.

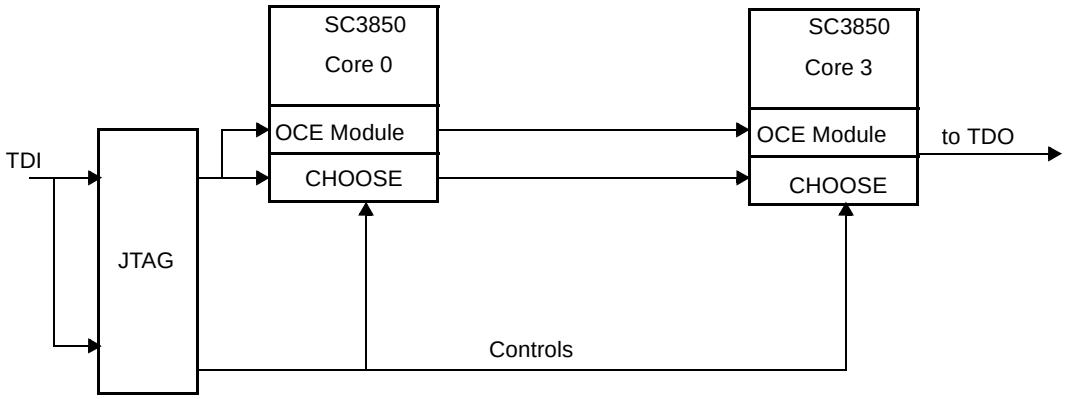


Figure 25-4. Cascading Multiple OCE Modules

Since all the OCE modules are cascaded, the selection procedure is performed serially. The sequence is:

1. Select the CHOOSE_ONCE instruction.
2. Enter at Shift_DR state the serial stream that specifies the modules to be selected (1 = selected, 0 = not selected).

The number of bits in this stream, that is, the number of clocks in this state, is equal to the number of selected SC3850 cores in the cascade, which is *six*. This state is indicated by the CHOOSE_CLOCK_DR signal. For example, for the six SC3850 cores on the MSC8156E, to activate the sixth core in the cascade, which is the closest to TDO and the farthest from TDI, the data is 1,0,0,0,0,0 (first a one, then five zeros). If the data is 0,0,1,0,1,0, then both the second and the fourth cells are selected.

Only the OCE command register (ECR) should be accessed in cascaded mode. To do this, first enter the CHOOSE_ONCE instruction and set ENABLE_ONCE to 1 for all cores. Then shift in the cascaded value for all ECRs in series. When the shift is ended and the controller issues a SHIFT_UPDATE, all registers are updated in parallel. However, it is not guaranteed that this occurs in the same SC3850 clock cycle for all cores.

25.1.6 DEBUG_REQUEST and ENABLE_ONCE Commands

After completing the CHOOSE_ONCE instruction, you can execute DEBUG_REQUEST and ENABLE_ONCE instructions. More than one such instruction can execute, and other instructions can be placed between them, as well as between them and the CHOOSE_ONCE instruction. The OCE modules selected in the CHOOSE_ONCE instruction remain selected until the next CHOOSE_ONCE instruction. The DEBUG_REQUEST or ENABLE_ONCE instruction is shifted in during the SHIFT-IR state, as are all JTAG instructions.

25.1.7 RD_STATUS Command

In the OCE, the status bits are no longer shifted out when shifting in any JTAG instruction. Instead, there is a special instruction which will return the status of selected core(s). **Figure 25-5** shows an example of CORE_CMD and RD_STATUS instructions.

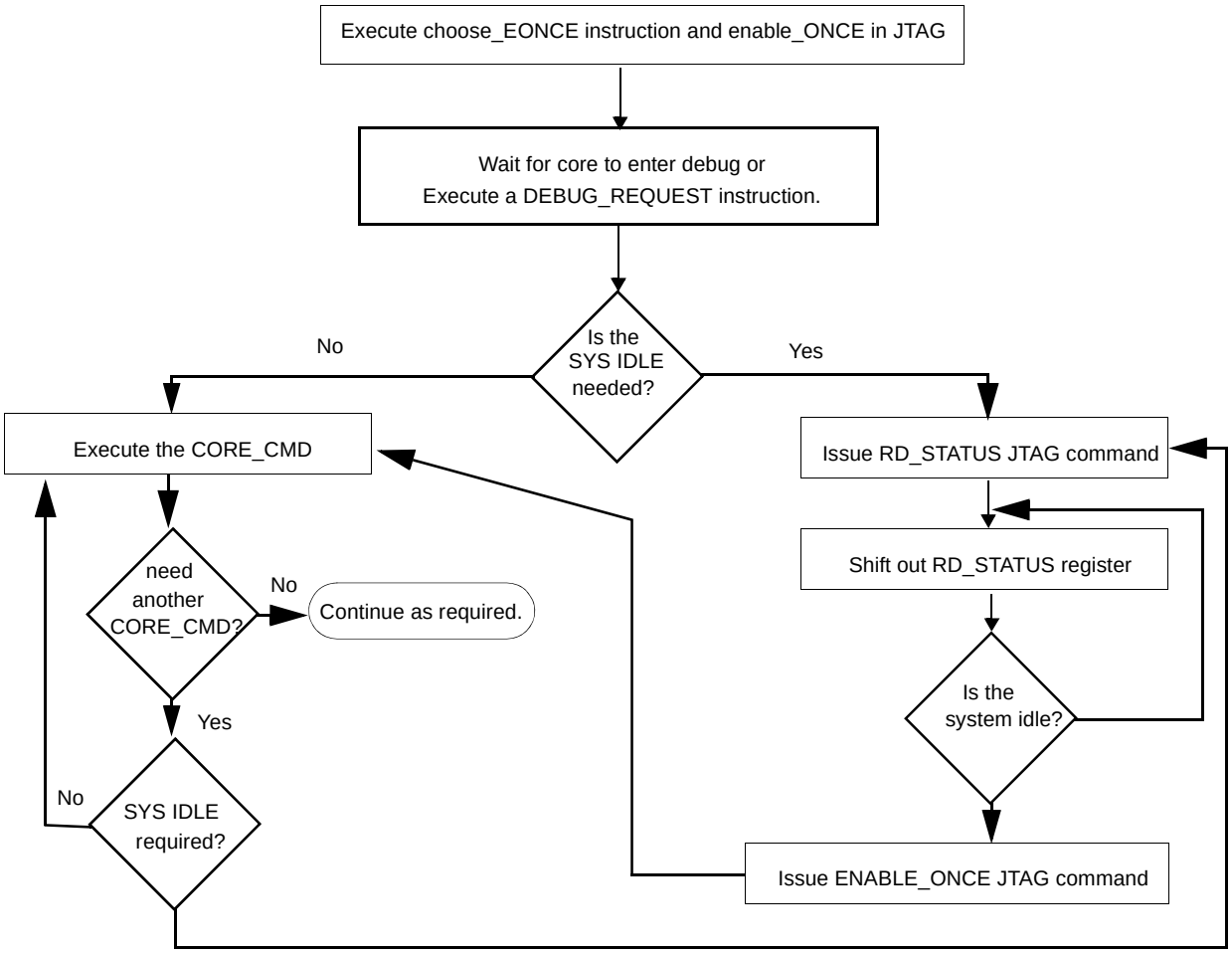


Figure 25-5. CORE_CMD and RD_STATUS Flow

25.1.8 Reading/Writing OCE Registers Through JTAG

An external host can read or write almost every OCE register through the JTAG interface by performing the following steps:

1. Execute the CHOOSE_ONCE command in the JTAG.
2. Send the data showing which OCE module is chosen. This command enables the JTAG to manage multiple OCE modules in a device.
3. Execute the ENABLE_ONCE command in the JTAG.

4. Write the OCE command into the ECR; that is, enter the JTAG TAP state machine into the SHIFT-DR state and then give the required command on the TDI input signal.

After the command is shifted in, the JTAG TAP state machine must enter the UPDATE-DR state. The data shifted via the TDI is sampled into the ECR. If, for example, the command written into the ECR is *Write EDCA0_CTRL*, then the host must again enter the JTAG into SHIFT-DR and shift the required data, which is to be written into the EDCA0_CTRL, via TDI. If the command is *read some register*, then the DR chain must be passed again and the contents of the register are shifted out through the TDO output signal. When JTAG shifts data to the OCE module, the lsb of the data is shifted first. See **Figure 25-6**.

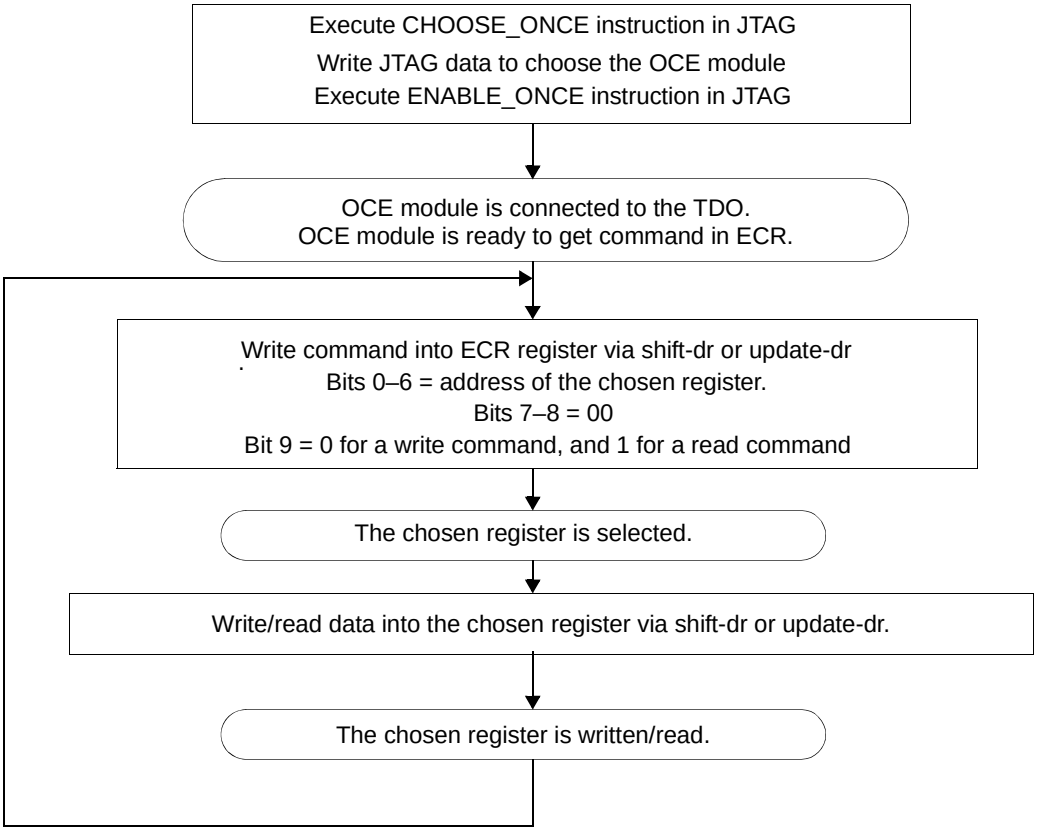


Figure 25-6. Reading and Writing OCE Registers Via the JTAG TAP

25.1.9 Signalling a Debug Request

The EE[0–1] signals connect to each of the MSC8156E OCE modules. EE0 is an input that signals a debug request; EE1 is an output signal that acknowledges the request or acts as an output of event detector 1.

Note: Asserting EE0 does not guarantee that the cores enter debug mode.The signal can be masked internally. Monitor the core status by shifting out the contents of PIREG or by issuing an instruction and observing the TDO value shifted out.

25.1.10 EE_CTRL Modifications for the MSC8156E

The relevant paragraph from the OCE module chapter of the *SC3850 DSP Core Reference Manual* is reproduced here with the appropriate amendments. See **Figure 7-20** in that manual for details.

EE_CTRL EE Control Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	—												EE1DEF		EE0DEF	
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The modes for EE[0–2] are restricted as follows:

Table 25-4. EE0 Definition (EE0DEF), Bits 1–0

EE0DEF		EE0 Definition
0	0	Reserved
0	1	Reserved
1	0	Input
1	1	Input: Debug Request

The EE0DEF bits program EE0, either to enable Address Event Detection Channel 0 by providing an input to the OCE module in the event detection unit and the event selector to or to generate an OCE event. EE0 can be programmed to enter the SC3850 core into Debug mode (the default) right after the SC3850 core is reset. Holding EE0 at logic value 1 during and after the reset puts the SC3850 core into Debug mode before the first dispatch occurs. In this mode, asserting EE0 also causes an exit from the STOP or WAIT processing states of the SC3850 core. If you want some SC3850 cores running and others in Debug mode, you must disable either the inputs of the running SC3850 cores or the outputs of the stopped SC3850 cores. To block EE0, set it as an input and mask the EE0 event in the event selector mask register (see the next section on programming the ESEL_DM Register).

Table 25-5. EE1 Definition (EE1DEF), Bits 3–2

EE1DEF		EE1 Definition
0	0	Output: Detection by EDCA1
0	1	Output: Debug Acknowledge
1	0	Reserved
1	1	Reserved

The EE1DEF bits program EE1. The signal can be programmed as an output of the OCE module to indicate detection by Address Event Detection Channel 1 (working as a toggle) or to indicate

that the DSP has entered Debug mode (Debug Acknowledge). To disable EE1, EDCA1 must be disabled and the mode set to 00.

Note: If the boot code is not executed, you must initialize the EE1DEF bits to 01 (output debug acknowledge) to activate EE1 as debug acknowledge. The default value (11) does not activate EE1 as debug acknowledge. If the EE1DEF bits are not initialized correctly, EE1 as a core output will always be 0, meaning that no debug acknowledge is sent to the other cores (as a signal to enter Debug mode) or to the EE1 output. Therefore, if the boot code is bypassed, the user must initialize EE1DEF correctly to use the debug acknowledge.

25.1.11 ESEL_DM and EDCA_CTRL Register Programming

The Event Selector Mask Debug Mode (ESEL_DM) register in the OCE programs the event selectors for the debug events. From the EE pins, the MSC8156E only supports EE0 and EE1 signals. Also, there is a requirement to block triggering from EE0 if only some SC3850 cores must enter Debug mode. The EE_CTRL register can be used to enable the use of EE1 as the debug indicator. See the *MSC8156 SC3850 DSP Core Subsystem Reference Manual* for more information.

25.1.12 Real-Time Debug Request

All the SC3850 cores can enter Debug mode in several ways. The EE0 debug input request of all six SC3850 cores is wired to the output of an “OR” gate that sums the state of all EE1 outputs of the other SC3850 cores and the external EE0 signal (see **Figure 25-7**). Therefore, if any one SC3850 core sets its EE1 output (that is, enters Debug mode) or EE0 is asserted, the debug request input on all SC3850 cores is asserted. EE1 is activated when at least one of the SC3850 cores enters Debug mode.

Note: The EE0 input initiates Debug mode, and the EE1 output is the debug acknowledge indication.

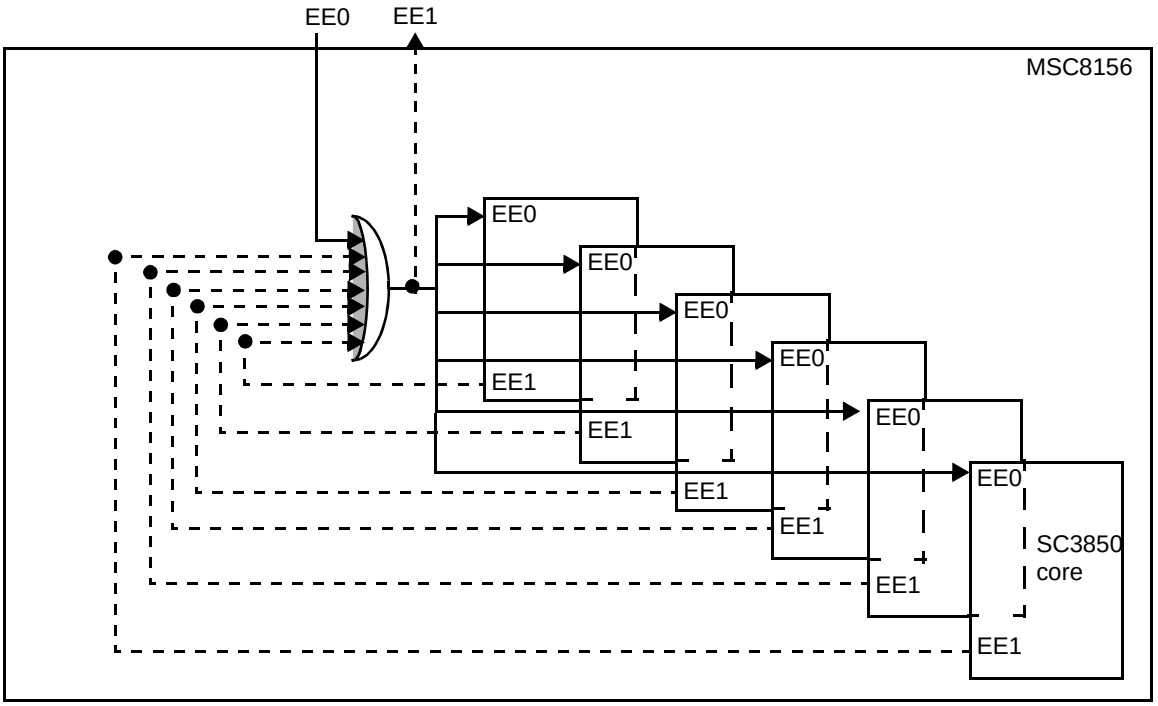


Figure 25-7. Selected SC3850 Core Issues a Debug Request to All Other SC3850 Cores

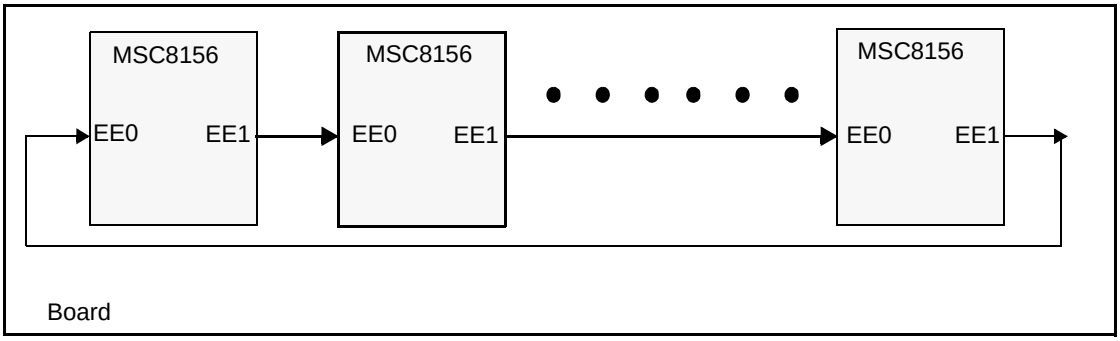


Figure 25-8. Board EE Signal Interconnectivity

25.1.13 Exiting Debug Mode

When an SC3850 core enters Debug mode (this is checked by OCE module status bits through JTAG as shown in **Table 25-2**), the EE0 internal signal of that SC3850 core OCE module is masked, preventing any more debug requests. When all the SC3850 cores exit Debug mode, the EE0 internal signals of all SC3850 cores are unmasked, enabling further debug requests. To restart the SC3850 cores, a **go** instruction is scanned into all six SC3850 cores. When the scan completes, the update launches all six SC3850 cores.

Note: When multiple cores are in Debug mode, issuing simultaneous **go** instructions to such cores does not guarantee that the cores exit Debug mode on the same clock cycle.

No retriggering occurs through EE0. For stepping, the same arrangement is used with the **step** instruction. All SC3850 cores are enabled via the CHOOSE_ONCE command, and then a **step** instruction is scanned into all six SC3850 cores. When the scan is done, the update launches all six SC3850 cores simultaneously. No retriggering occurs through EE0.

25.1.14 General JTAG Mode Restrictions

The control afforded by the output enable signals using the **bsr** and the **extest** instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. You must avoid situations in which the MSC8156E output drivers are enabled into actively driven networks. There are two constraints on the JTAG interface.

- The TCK input does not include an internal pull-up resistor. To preclude mid-level input effects, do not leave this line unconnected.
- To ensure that the JTAG logic does not conflict with the system logic, always force the TAP into the test-logic-reset controller state by asserting the $\overline{\text{TRST}}$ input during power-up.

To save power when JTAG is not in use, the MSC8156E should be in the following state:

- To enter or to remain in the Low-Power Stop mode, the TAP controller must be in the test-logic-reset state. Leaving the TAP controller test-logic-reset state negates the ability to achieve low power but does not otherwise affect device functionality.
- The TCK input is not blocked in Low-Power Stop mode. To consume minimal power, the TCK input should externally connect to V_{CC} or ground.
- TMS and TDI include internal pull-up resistors. In Low-Power Stop mode, these two signals should remain either unconnected or connected to V_{CC} to achieve minimal power consumption.

25.1.15 JTAG and OCE Module Programming Model

25.1.15.1 Identification Register

The JTAG ID register is a 32-bit read-only factory-programmed register that distinguishes the component on a board according to the **IEEE** Std. 1149.1. The fields are defined as follows:

JTAGID JTAG Identification (ID) Register **JTAG port access only**

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
	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1
	Version				Design Center						Sequence Number										Manufacturer identity											1

- Version information corresponds to the revision number. The MSC8156E first mask set is 0b0000.
- Design Center number is 0b000110.
- Sequence Number for the MSC8156E is 0b0010010100.
- Manufacture identity is 0b00000001110.
- The final 1 is required by the **IEEE** Std. 1149.1.

Note: The hexadecimal value stored in this register is 0x0189501D.

Later mask sets will have a different number. Refer to the website listed on the back cover of this manual for the information about the contents of this register for current device revisions.

25.1.15.2 Boundary Scan Register (BSR)

The MSC8156E BSR contains bits for most device signals and control signals. All MSC8156E bidirectional signals have two registers for boundary scan data and are controlled by an associated control bit in the BSR. The boundary scan bit definitions vary according to the specific chip implementation of the MSC8156E and are described by the BSDL file on the product website. **Figure 25-9** through **Figure 25-12** show various BSR cell types.

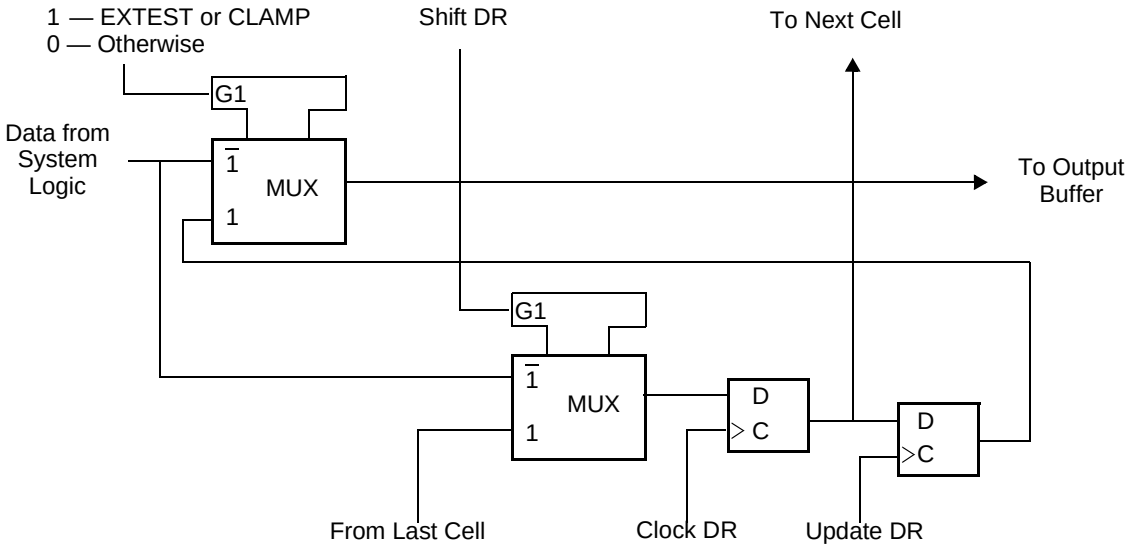


Figure 25-9. Output Signal Cell (O.PIN)

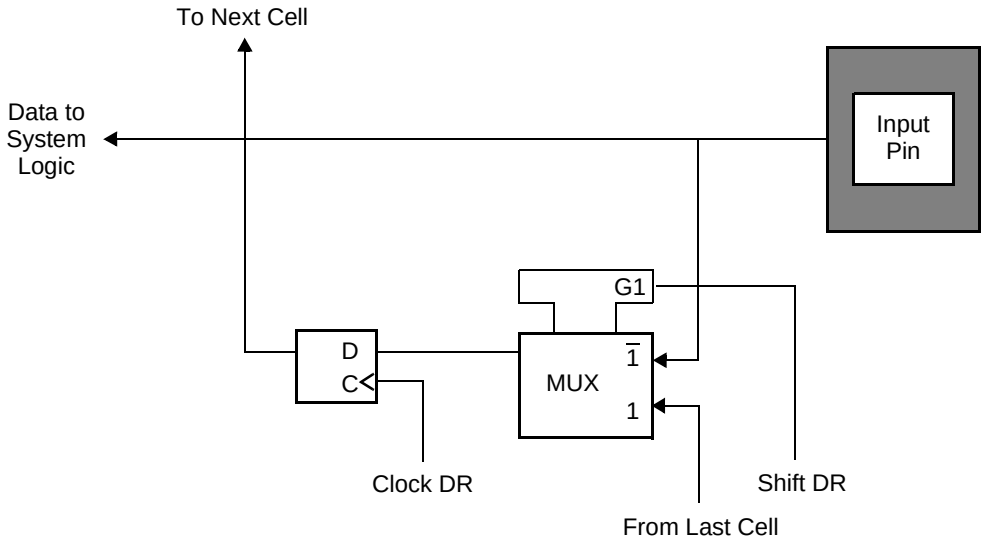


Figure 25-10. Observe-Only Input Signal Cell (I.OBS)

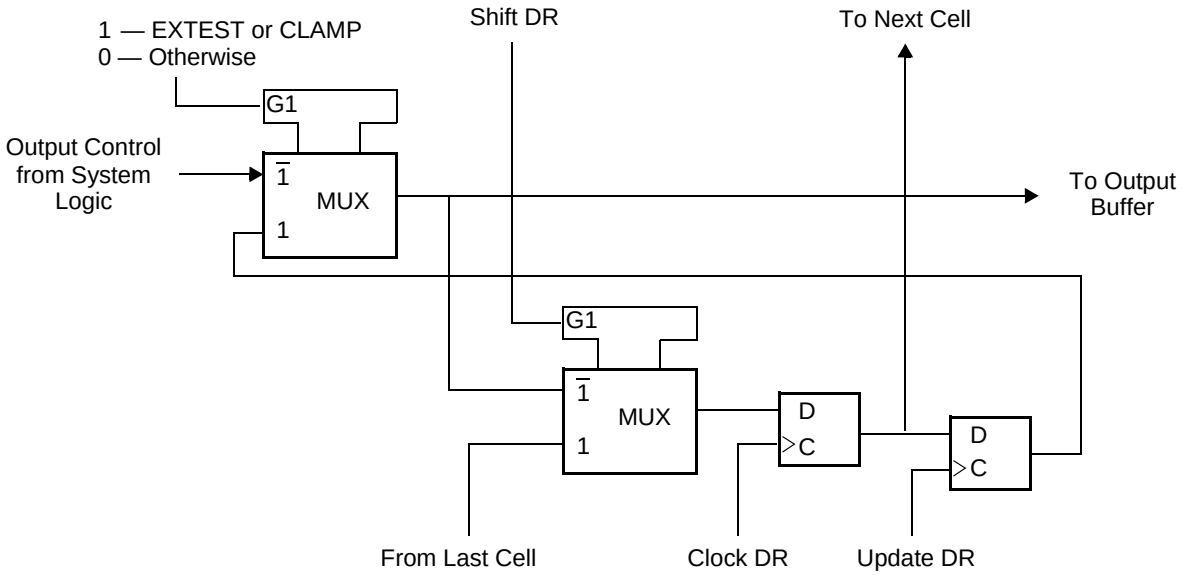


Figure 25-11. Output Control Cell (IO.CTL)

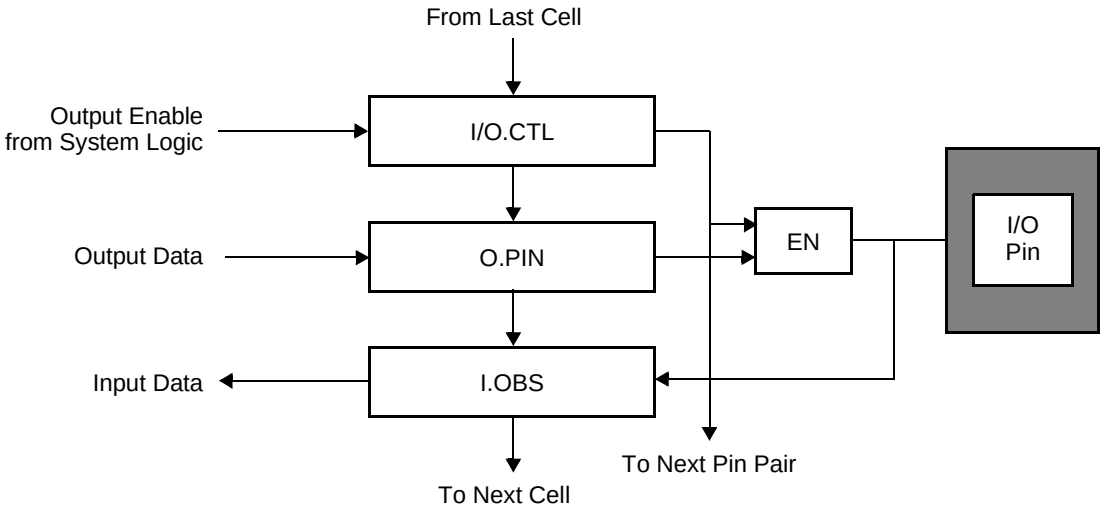


Figure 25-12. General Arrangement of Bidirectional Signal Cells

The control bit value controls the output function of the bidirectional signal. One or more bidirectional data cells can be serially connected to a control cell. Bidirectional signals include two scan cells for data (IO.Cell) as shown in **Figure 25-12**, and these bits are controlled by the cell shown in **Figure 25-11**. It is important to know the boundary scan bit order and signals that are associated with them. The BSDL file on the product website describes the boundary sca serial string. The three MSC8156E cell types described in this file are depicted in **Figure 25-9** through **Figure 25-11**, which describe the cell structure for each type.

25.1.15.3 Shift Registers

The shift registers include the Bypass Register, General-Purpose Register (GPR), BSR, Identification Register, and Parallel Input register.

25.1.15.4 Bypass Register

The Bypass Register is a single-bit shift register (see **Figure 25-13**). When selected, it creates a shift-register path of one bit from TDI to TDO. When the Bypass Register is selected, the shift-register stage is set to a logic zero on the rising edge of TCK in the CAPTURE-DR controller state.

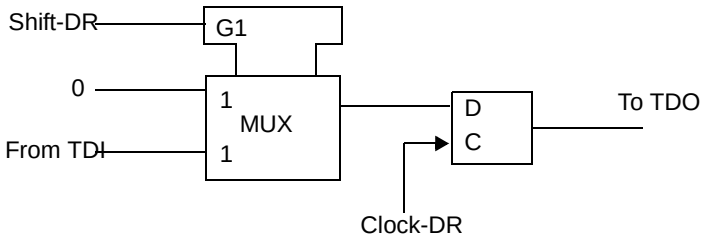


Figure 25-13. Bypass Register Configuration

25.1.15.5 Identification Register

When the Identification Register is selected, the shift-register stage is set to a logic value equal to IDCODE on the rising edge of TCK in the CAPTURE-DR controller state. It can then be shifted out in the SHIFT-DR controller state. See **Figure 25-14**.

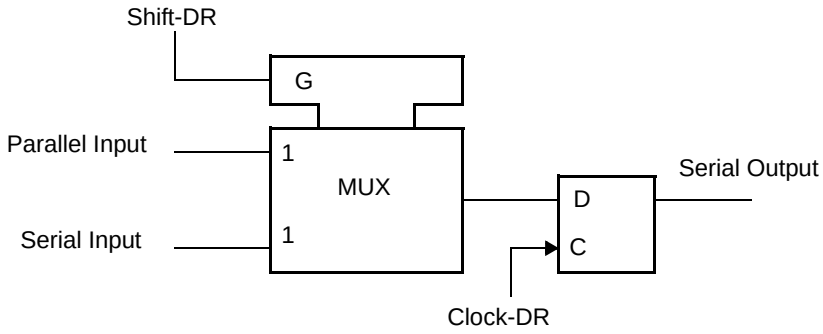


Figure 25-14. Identification Register Configuration (ID)

25.2 Debug and Profiling

The main debugging and profiling purposes are:

- Parallel counting of different events characterizing the operation of the MSC8156E device.
- Support for entering Debug mode upon detecting a predefined state of the MSC8156E device, for example, when counting a predefined number of events (watchpoint monitors).
- Support for tracing of various modes in the QUICC Engine module and DSP core subsystem.
- Debug errors (for example, a transaction request with an illegal address or peripherals errors).
- Support of reading and writing all MSC8156E registers and memories in debug mode by a host processor.

25.2.1 Features

Table 25-6 describes MSC8156E device debug and profiling features

Table 25-6. MSC8156E Debug and Profiling Features

Block Name	Number of Blocks in MSC8156E	Supports Internal Debug	Supports Internal Profiling	Supports Profiling by PM Block	Supports Profiling and Debug by a CLASS Module
DSP core subsystem	6	+	+	+	+
DMA	1	+	—	+	+
QUICC Engine subsystem	1	+	+	—	+
Class	2	+	+	—	+
TDM	4	+	—	—	+
PCI	1	+	—	—	+
RapidIO complex	1	+	—	+	+
L2/M2 memory	6	—	—	—	+
M3 memory	1	—	—	—	+
DDR memory	1	—	—	—	+

Other features:

- The performance monitor block supports counting of RapidIO and DMA specific events.
- The watchdog timer (WDT) option prevents system lock if software becomes trapped in a loop operation with no controlled exit.
- JTAG port
  - Support multiple core OCE control and interconnection for the DSP core subsystems.
  - Supports QUICC Engine module debug control.
  - Provides access to all shared and CCSR memory space.
- The MSC8156E provides access for all peripherals (QUICC Engine subsystem, RapidIO, PCI Express, TDM, DMA, and UART) to all shared and CCSR memory space.

25.2.2 Entering Debug Mode

The individual modules have their own Debug state, which can be entered as follows:

- *DSP core subsystem*. Enters the Debug state when one of the following events occur:
  - Assertion of dedicated input signals (normally connected to the debugging agent)
  - Execution of the DEBUG or DEBUGEV instructions by the core.
  - A DPU event (depends on the configuration of the DPU and OCE).
  - Initiator or peripheral writes a certain value to the GCR2 control register.
- *L1 ICache and DCache and L2 Cache*. Activated only when the DSP core subsystem is in the Debug state and certain values are written to their respective control registers. In this mode, the internal state of the caches (tags, valid bits, PLRU table and cache array) can be read with JTAG-inserted core commands.

Note: See *SC3850 DSP Core Reference Manual* and *MSC8156 SC3850 DSP Core Subsystem Reference Manual* for details. Both are available under NDA. Contact your local Freescale sales office or representative for details.

25.2.3 Exiting Debug mode

The modules with a Debug mode exit that mode as follows:

1. The ICache, DCache and L2 Cache blocks exit the Debug state when certain values are written to their respective control registers through the JTAG port or a reset signal is asserted.
2. The SC3850 DSP core subsystems exit the Debug state when they receive the proper transaction from the external debugging agent through the JTAG port, or a reset signal is asserted.

25.2.4 SC3850 Debug and Profiling

The MSC8156E device contains six extended DSP cores. Each DSP core subsystem supports the debug and profiling capabilities. When the DSP core subsystem is in the Debug state, the SC3850 core enters its Debug processing state, and instruction processing is halted. After a delay, all subsequent DSP core subsystem activity ceases (as reflected in the BUSY bit in the JTAG accessible OCE register RD_STATUS). In this state, a debugging agent external to the DSP core subsystem can access various internal DSP core subsystem registers and memory locations to develop and debug applications. The DSP core subsystem enters Debug state after one of the following occurs:

- Assertion of dedicated input signals (normally connected to the debugging agent).
- Execution of the DEBUG or DEBUGEV instruction by the core.
- An event is detected by the DPU (depending on the configuration of the DPU and OCE).
- An initiator or peripheral device writes a certain value to GCR2 control register.

Note: See the *MSC8156 SC3850 DSP Core Subsystem Reference Manual* for details.

The DSP core subsystem exits the Debug state when it receives the proper transaction from the external debugging agent through the JTAG port or a reset signal is asserted.

25.2.5 L1 ICache and DCache Debug and Profiling

The L1 ICache and DCache L2 Cache/M2 blocks in each DSP core subsystem have block-specific Debug modes that are ctivated only when the DSP core subsystem is in the Debug state and certain values are written to their respective control registers. In this mode, the internal state of the caches (tags, valid bits, PLRU table and cache array) can be read with JTAG-inserted core commands.

Note: See the *MSC8156E MSC3850 Core Subsystem Reference Manual* for details.

25.2.6 DMA Controller Debug and Profiling

The DMA controller can enter debug mode only as the result of an external debug request. When this occurs, the channel logic masks all channel requests generated towards the bus interface and finishes all pipelined requests in the bus interface and M bus. In this state, a debugging agent external to the MSC8156E can access the DMA controller PRAM through JTAG bus.

25.2.6.1 Debug Errors

The DMA support debugging errors and indications, such as:

- BD_SIZE, MD_BD_SIZE programmed with a value of zero
- Channel information that causes an illegal addresses on bus interface ports A/B.
- Early Dead Line serve First Violation.

When one of the errors occurs, the DMA controller generates the error interrupt listed in **Table 25-7**.

Table 25-7. DMA Debug Interrupt

DSP core subsystem Interrupt Number	Description of Interrupt
143	DMA errors interrupt

See **Chapter 14**, *Direct Memory Access (DMA) Controller* for details.

25.2.6.2 Profiling Unit

The 16 channel DMA controller supports system level profiling. You can profile specific channels by configuring the desired channel number (CHA_NUM) and channel source or destination (DEST) fields in the DMA_LPCR. See **Section 14.5**, *Profiling* for details.

After configuration, all DMA events are connected to and monitored by the performance monitor (PM) block. See **Section 25.3**, *Performance Monitor* for details on how to use the information.

25.2.7 CLASS Modules

Each of the CLASS modules includes the ability to generate interrupts and perform profiling.

25.2.7.1 Debug

Each CLASS module can generate up to N + 1 interrupts, which are divided to 2 groups: N particular interrupts (one per MI M bus Initiator) and one general Interrupt. A specific interrupt is created when the CLASS module receives a transaction request with an illegal address. Illegal addresses are defined as one of the following two cases:

1. An address that does not belong to any of the address space windows of the enabled address decoders.
2. An address that falls within any of the address space windows of the enabled error address decoders.

The general interrupt is the logical OR of all the particular interrupts. Thus, the general interrupt is asserted when at least one of the particular interrupts is asserted.

Note: See **Chapter 4**, *Chip-Level Arbitration and Switching System (CLASS)* for details.

25.2.7.2 CLASS Debug Profiling Unit (CDPU)

The CLASS supports Debug and Profiling measurements by the class debug profiling unit (CDPU) sub-block. The main features are:

- Time-out mechanism. This mechanism does not generate an interrupt. The host must poll certain a bit in the respective CLASS register.
- Watch point mechanism profiling unit. The CLASS profiling unit provides the following profiling information:
  - Data acknowledges of write accesses.
  - Data acknowledges of read accesses.
  - Acknowledged accesses (req and req_ack).
  - Stall cycles due to write-after-read.
  - Cycles of non-acknowledged accesses.
  - Acknowledged supervisor accesses.
  - Acknowledged non-supervisor accesses.
  - Cycles when priority = 0.
  - Cycles when priority = 1.
  - Cycles when priority = 2.
  - Cycles when priority = 3.
  - Priority upgrades.
  - Cycles for which the priority was not upgraded because the upgradeable signal was low.
  - Acknowledged read accesses.
  - Acknowledged write with confirm accesses.
  - Acknowledged write without confirm accesses.
- Over-flow mechanism.

Table 25-8. Class0 Debug Interrupts

DSP Core Subsystem Interrupt	Description of Interrupt
Routed through General Interrupt Register 3 (GIR3)	Class0 watch point mechanism interrupt
Routed through General Interrupt Register 3 (GIR3)	Class0 over flow mechanism interrupt
Routed through General Interrupt Register 3 (GIR3)	Class0 error interrupt

Chapter 4, *Chip-Level Arbitration and Switching System (CLASS)* and **Chapter 8**, *General Configuration Registers* for details.

25.2.8 QUICC Engine Debug and Profiling

The QUICC Engine module has several means to debug and profile its operation as described in the following sections.

25.2.8.1 Trace Buffer

The QUICC Engine module provides chip-level testing capability through the trace buffer block (TRB). The TRB provides means of tracing the code ran by the CP (communication processor) in real time (through storing it non-intrusively) and reporting several internal CP/TRB events. The QUICC Engine module RISC has 4 kinds of breakpoints for on chip software debugging

- Instruction breakpoint
- Software breakpoint
- Data breakpoint
- External breakpoint

Note: See **Section 18.2**, *RISC Processors* for details.

25.2.8.2 Loopback Modes

SGMII supports an internal Loopback mode in the TBI MAC layer. RGMII supports internal Loopback mode in the MAC layer. The communication controllers support Diagnostic modes, including local and external loopback (transmitter-to-receiver) and normal operation. See **Section 19.4**, *Diagnostic Modes* for details.

The Serial RapidIO controller supports analog and digital loopback mode. See **Chapter 16**, *Serial RapidIO Controller* for details.

25.2.9 TDM Debug and Profiling

The TDM modules provide a set of debug interrupts and loopback support for debugging.

25.2.9.1 Debug

The TDM complex in the MSC8156E device consists of four TDM modules. Each TDM module supports transmit sync error and receive sync error interrupts. **Table 25-9** lists the TDM debug interrupts.

Table 25-9. TDM Debug Interrupts

DSP core subsystem Interrupt Number	Description of Interrupt
Routed through General Interrupt Register 1 (GIR1)	RX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	TX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	RX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	TX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	RX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	TX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	RX sync error interrupt
Routed through General Interrupt Register 1 (GIR1)	TX sync error interrupt

See **Chapter 21**, *TDM Interface* and **Chapter 8**, *General Configuration Registers* for details.

25.2.9.2 TDM Loopback Support

The TDM modules support loopback test mode in which the receiver receives the same data that is transmitted out. See **Section 21.5**, *Loopback Support* for details.

25.2.10 RapidIO Debug and Profiling

The RapidIO system debugging system includes an error interrupt and the ability to connect to the profiling unit.

25.2.10.1 Debug Errors

The RapidIO system asserts the error interrupt listed in **Table 25-10** when a system error is detected.

Table 25-10. RapidIO Debug Interrupt

DSP core subsystem Interrupt Number	Description of Interrupt
90	RapidIO Errors interrupt

See **Chapter 16**, *Serial RapidIO® Controller* for details.

25.2.10.2 Profiling Unit

All RapidIO events can connect to Performance monitor (PM) block. See **Section 25.3**, *Performance Monitor* for details. The RapidIO PM uses the RapidIO clock.

25.2.11 MAPLE-B Debug

The MAPLE-B subsystem supports an error interrupt. If any of these errors occur, the interrupt is asserted.

Table 25-11. MAPLE-B Debug Interrupts

DSP core subsystem Interrupt	Description of Interrupt
Routed through General Interrupt Register 3 (GIR3)	MAPLE-B Errors interrupt

See **Chapter 28**, *Multi Accelerator Platform Engine, Baseband (MAPLE-B)* and **Chapter 8**, *General Configuration Registers* for details.

25.2.12 Software Watchdog (SWT)

The watchdog timer (WDT) option prevents system lockup in cases where the software becomes trapped in a loop with no controlled exit. Watchdog timer operations are configured in the System Watchdog Control Register (SWCRR). See **Chapter 8**, *General Configuration Registers* for details.

25.2.13 Profiling Unit Programming Model

All DPU registers are memory-mapped and can be written or read by the core in Execution mode or through the OCE Core Command in Debug mode. Only one access per execution set to a DPU register is allowed in order to assure that the programming of the DPU registers occurs in a deterministic order. Reserved or unused bits in all registers should be written as zeros and the read value should be masked. Writing to unimplemented or read-only registers has no effect, and should be avoided for future software compatibility. Reading from unimplemented or write-only registers is illegal and produces undefined results.

Writing and reading of the DPU registers is done via the QBus. The DPU registers are located in Bank 0. This means that there are a number of cycles until the value is actually written to the DPU registers. In case DPU register synchronization is important, then the programming of the last DPU register should have a SYNCIO instruction in parallel. Code following this action can assume that the DPU registers written earlier were actually written. All registers are 32 bits with 16-bit accesses, which enable the use of bit-mask operations. When a 16-bit access is used on the 32-bit registers, the software address offset to the MSB part of the registers is equal to the software address offset of the LSB part + 2. The LSB part of the address is as shown in the registers memory map, and is not influenced by whether the system is Big Endian or Little

Endian.) Any other access type other than word or long (such as byte, 2 long) should be avoided, and will result in an undefined result. When accessing the counter value registers, the 31 LSBs of the bus are written to the 31 LSBs of the register. Bit 31 of these registers is reserved.

- General Registers
  - DPU Control Register (DP_CR)
  - DPU Status Register (DP_SR)
  - DPU Monitor Register (DP_MR)
  - DPU PID Detection Reference Value Register (DP_RPID)
  - DPU DID Detection Reference Value Register (DP_RDID)
- General Counters
  - DPU Counter Triad A Control Register (DP_TAC)
  - DPU Counter Triad B Control Register (DP_TBC)
  - DPU Counter A0 Control Register (DP_CA0C)
  - DPU Counter A0 Value Register (DP_CA0V)
  - DPU Counter A1 Control Register (DP_CA1C)
  - DPU Counter A1 Value Register (DP_CA1V)
  - DPU Counter A2 Control Register (DP_CA2C)
  - DPU Counter A2 Value Register (DP_CA2V)
  - DPU Counter B0 Control Register (DP_CB0C)
  - DPU Counter B0 Value Register (DP_CB0V)
  - DPU Counter B1 Control Register (DP_CB1C)
  - DPU Counter B1 Value Register (DP_CB1V)
  - DPU Counter B2 Control Register (DP_CB2C)
  - DPU Counter B2 Value Register (DP_CB2V)
- Trace Buffer Registers
  - DPU Trace Control Register (DP_TC)
  - DPU VTB Start Address Register (DP_TSA)
  - DPU VTB End Address Register (DP_TEA)
  - DPU Trace Event Request Register (DP_TER)
  - DPU Trace Write Pointer Register (DP_TW)
  - DPU Trace Data Register (DP_TD)

Note: The DPU registers use the base address: 0xFFF0A000.

25.2.13.1 DPU Control Register (DP_CR)

DP_CR DPU Control Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		TIDCM		ISEDCA5		ISEDCA4		ISEDCA3		ISEDCA2		ISEDCA1		ISEDCA0	
Type	R/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
	—	EIS	DETB		DECB2		DECB1		DECB0		DECA2		DECA1		DECA0	
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CR register is a 32-bit register responsible for controlling the debug logic of the DPU and the task ID comparator. **Table 25-12** defines the DP_CR bit fields.

Table 25-12. DP_CR Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TIDCM 29–28	0	**Task ID Comparator Mask** Controls the operation of the task ID comparator, deciding which part takes part in the comparison. The reference program and data ID values are written in the DP_RPID and DP_RDID registers.	00 Neither the data task ID or the program task ID participate in the comparison. The comparison result is always 1. 01 The data task ID does not participate in the comparison (masked). 10 The program task ID does not participate in the comparison (masked). 11 Both the program task ID and the data task ID participate in the comparison.
ISEDCA5 27–26	0	**Interrupt Selector EDCA5** An event generated by the EDCA5 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA4 25–24	0	**Interrupt Selector EDCA4** An event generated by the EDCA4 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA3 23–22	0	**Interrupt Selector EDCA3** An event generated by the EDCA3 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA2 21–20	0	**Interrupt Selector EDCA2** An event generated by the EDCA2 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC

Table 25-12. DP_CR Bit Descriptions (Continued)

Name	Reset	Description	Settings
ISEDCA1 19–18	0	**Interrupt Selector EDCA1** An event generated by the EDCA1 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
ISEDCA0 17–16	0	**Interrupt Selector EDCA0** An event generated by the EDCA0 channel of the OCE causes interrupt Debug A or Debug B to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
— 15	0	Reserved. Write to zero for future compatibility.	
EIS 14	0	**OCE Interrupt Selector** A maskable interrupt generated by the OCE is directed either to interrupt Debug A or Debug B.	0 A maskable interrupt generates Debug A 1 A maskable interrupt generates Debug B
DETB 13–12	0	**Trace Buffer Debug Request/Interrupt Enable** An event generated by the trace logic of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECB2 11–10	0	**Counter B2 Debug Request/Interrupt Enable** An event generated by counter B2 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECB1 9–8	0	**Counter B1 Debug Request/Interrupt Enable** An event generated by counter B1 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECB0 7–6	0	**Counter B0 Debug Request/Interrupt Enable** An event generated by counter B0 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECA2 5–4	0	**Counter A2 Debug Request/Interrupt Enable** An event generated by counter A2 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECA1 3–2	0	**Counter A1 Debug Request/Interrupt Enable** An event generated by counter A1 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC
DECA0 1–9	0	**Counter A0 Debug Request/Interrupt Enable** An event generated by counter A0 of the DPU causes a debug request to the OCE or an interrupt to the EPIC.	00 Does not cause an interrupt 01 reserved 10 Generates Debug A interrupt to the EPIC 11 Generates Debug B interrupt to the EPIC

25.2.13.2 DPU Status Register (DP_SR)

DP_SR DPU Status Register Offset 0x04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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
	—									TWBA	ENCB2	ENCB	ENCB0	ENCA2	ENCA1	ENCA0
Type	R															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_SR is a 32-bit register that reflects the status of the DPU counters and if a trace write buffer (TWB) flush is in progress. All the bits are sticky status bits, and are read-only. Only the 6 lsb bits are used to enable the execution of bit-mask instructions.

Note: The ENCA and ENCB bits are used whenever the counter is enabled or disabled by events that occur after the DPU is initialized (for example, an EDCA0 event enables the counter and the DEBUGEV instruction disables it).

Table 25-13 defines the DP_SR bit fields.

Table 25-13. DP_SR Bit Descriptions

Name	Reset	Description	Settings
— 31–7	0	Reserved. Write to zero for future compatibility.	
TWBA 6	0	**Trace Write Buffer Active** The user can poll this bit to determine whether to disable tracing by setting the DP_TC[TMPDIS] bit or by clearing the DP_TC[EN] bit.	0 Trace Write Buffer flush is complete. 1 Flush sent to Trace Write Buffer.
ENCB2 5	0	**Counter B2 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCB1 4	0	**Counter B1 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCB0 3	0	**Counter B0 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCA2 2	0	**Counter A2 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.
ENCA1 1	0	**Counter A1 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.

Table 25-13. DP_SR Bit Descriptions (Continued)

Name	Reset	Description	Settings
ENCA0 0	0	**Counter A0 Enable** Indicates whether the counter is disabled or enabled.	0 Disabled. 1 Enabled.

25.2.13.3 DPU Monitor Register (DP_MR)

DP_MR DPU Status Register Offset 0x08

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—															
Type	R															
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
	—						TBF	DRA	—	DRTB	DRCB2	DRCB1	BRCB0	DRCA2	DRCA1	DRCA0
Type	R						W1C									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_MR is a 32-bit register that reflects information about an event generated by a counter or the trace buffer. All the bits are sticky bits, and can be only cleared by writing 1 to the appropriate bit(s). Only the 16 lsbs are used to enable the execution of bit-mask instructions. **Table 25-14** defines the DP_SR bit fields.

Table 25-14. DP_MR Bit Descriptions

Name	Reset	Description	Settings
— 31–10	0	Reserved. Write to zero for future compatibility.	
TBF 9	0	**Trace Buffer Finished** Indicates whether the trace buffer has written the last entry in the VTB.	0 Trace buffer has not written the last entry to the VTB. 1 Trace buffer has written the last entry to the VTB.
DRA 8	0	**Debug is External Asynchronous Debug Request** Indicates whether an external synchronous debug request.	0 No external synchronous debug request. 1 External synchronous debug request.
— 7	0	Reserved. Write to zero for future compatibility.	
DRTB 6	0	**Debug/Interrupt Reason is Trace Buffer** Indicates a trace buffer event debug request.	0 No trace buffer event debug request. 1 Trace buffer event debug request.
DRCB2 5	0	**Debug/Interrupt Reason is Counter B2 Event** Indicates a counter B2 Event debug request.	0 No counter B2 event debug request. 1 Counter B2 event debug request.
DRCB1 4	0	**Debug/Interrupt Reason is Counter B1 Event** Indicates a counter B1 Event debug request.	0 No counter B1 event debug request. 1 Counter B1 event debug request.

Table 25-14. DP_MR Bit Descriptions (Continued)

Name	Reset	Description	Settings
DRCB0 3	0	**Debug/Interrupt Reason is Counter B0 Event** Indicates a counter B0 Event debug request.	0 No counter B0 event debug request. 1 Counter B0 event debug request.
DRCA2 2	0	**Debug/Interrupt Reason is Counter A2 Event** Indicates a counter A2 Event debug request.	0 No counter A2 event debug request. 1 Counter A2 event debug request.
DRCA1 1	0	**Debug/Interrupt Reason is Counter A1 Event** Indicates a counter A1 Event debug request.	0 No counter A1 event debug request. 1 Counter A1 event debug request.
DRCA0 0	0	**Debug/Interrupt Reason is Counter A0 Event** Indicates a counter A0 Event debug request.	0 No counter A0 event debug request. 1 Counter A0 event debug request.

25.2.13.4 DPU PID Detection Reference Value Register (DP_RPID)

DP_RPID DPU PID Detection Reference Value Registers Offset 0x0C

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

The DP_RPID is a 32-bit register containing the reference program ID value for the task ID comparator. The DPU Control register controls the operation mode of the comparator. **Table 25-15** defines the DP_RPID bit fields.

Table 25-15. DP_RPID Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RPID 7–0	0	**Reference Program ID Value** Stores the value of the reference program ID.	

25.2.13.5 DPU DID Detection Reference Value Register (DP_RDID)

DP_RDID DPU DID Detection Reference Value Registers Offset 0x10

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

The DP_RDID is a 32- bit register containing the reference data ID value for the task ID comparator. The DPU Control register controls the operation mode of the comparator. **Table 25-16** defines the DP_RDID bit fields.

Table 25-16. DP_RDID Bit Descriptions

Name	Reset	Description	Settings
— 31–8	0	Reserved. Write to zero for future compatibility.	
RDID 7–0	0	**Reference Data Task ID Value** Stores the value of the reference data task ID.	

25.2.13.6 DPU Counter Triad A Control Register (DP_TAC)

DP_TAC DPU Triad A Control Register Offset 0x20

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		TDMP		TDM				—		TENMP		TENM			
Type	R/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
	—		CEGP		—			CEG					—	CMODE		TCEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_TAC register is a 32-bit register that controls the operation of the DPU counter triad A, including what events and when they are counted. If the TCEN bit in the DP_TAC register is set, the appropriate counters are controlled by this register and ignore the programming of their own control register. When the TCEN bit is cleared, each counter is controlled individually by its own control register. **Table 25-20** defines the DP_TAC bit fields.

Table 25-17. DP_TAC Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TDMP 29–28	0	**Triad Disable Mode Privilege Level** The event disabling the counters belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TDM 27–24	0	**Triad Disable Mode** The event that disables the counters in the triad.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
TENMP 21–20	0	**Triad Enable Mode Privilege Level** The event enabling the counters belongs to the task described by these bits. If the MARK instruction enables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TENM 19–16	0	**Triad Enable Mode** The event that enables the counters.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-17. DP_TAC Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEGP 13–12	0	**Counted Event Group Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CEG 8–4	0	**Counted Event Group** The source counted by the counter.	See **Table 25-18** for details.
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Triad Counters Mode** Specifies the mode of the counter	00 One shot. Each counters in the triad generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. Each counter in the triad has its value saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
TCEN 0	0	**Triad Control Enable** Determines whether the three counters in the triad are controlled by this control register or their individual control counters.	0 Each counter is controlled independently. 1 The three counters are controlled by this register and the individual register settings have no effect.

Table 25-18. Counted Event Group

CEG Value	Group Name	Counter 0 Counts	Counter 1 Counts	Counter 2 Counts
00000	ICache hit-miss	ICache misses (without prefetch hits)	ICache hits	ICache prefetch hits
00001	DCache hit-miss	DCache misses (without prefetch hits)	DCache hits	DCache prefetch hits
00010	Core wait state	Clock cycles (non-debug)	Wait processing-state cycles	Not used
00011	Breakdown of application cycles - Group 1	Application cycles (non-debug, non-wait, non-stop)	No bubble	Bubble due to COF or interrupt
00100	Breakdown of application cycles - Group 2	Bubble due to starvation (no instructions in the prefetch/dispatch buffer)	Bubble due to core resource conflicts	Bubble due to data memory holds
00101	Not implemented in the MSC8156E			
00110	Not implemented in the MSC8156E			

Table 25-18. Counted Event Group (Continued)

CEG Value	Group Name	Counter 0 Counts	Counter 1 Counts	Counter 2 Counts
00111	Hold associated with debug	Hold due to VTB writes	Hold due to Nexus freeze	not used
01000	Hold due to WRQ or WTB	Hold due to WRQ flush or atomic operation	Hold due to WRQ hazard	Hold due to WRQ or WTB full
01010	Hold due to Dcache system	Hold due to cacheable access (read, write-back miss or write-trough prefetch hit).	Hold due to non-cacheable access (read)	not used
01011	Not implemented in the MSC8156E			
01100	Master bus No. 1 load (See Section 10.2.1.8.8)	Master bus No. 1 bus cycles	Instruction transfer cycles on Master bus No. 1	Data transfer cycles on Master bus No. 1
01101	Master bus No. 2 load (See Section 10.2.1.8.8)	Master bus No. 2 bus cycles	Instruction transfer cycles on Master bus No. 2	Data transfer cycles on Master bus No. 2
01110	Bus load due to VTB write	Master bus No. 1 bus cycles	Data transfer to the VTB (TWB write accesses)	Master bus No. 2 bus cycles
10000	Instruction accesses to L2 subsystem	L2 instruction access miss	L2 instruction access hit	Total L2 instruction accesses
10001	Data accesses to L2 subsystem	L2 data access miss	L2 data access hit	Total L2 data accesses
10010	M2 RAM contentions (if L2 subsystem exists)	Contentions between IQ DQ accesses	Contentions between Q (IQ or DQ) and DMA accesses	not used
10011	M2 ROM contentions (if L2 subsystem exists)	Contentions between IQ DQ accesses	Contentions between Q (IQ or DQ) and DMA accesses	not used
10100	BTB Characterization Group 1	Total number of execution sets	Sequentially executed execution sets.	BTB-able instructions correctly predicted
10101	BTB Characterization Group 2	BTB-able instructions not in the BTB, wrongly predicted.	BTB-able instructions, in the BTB, wrongly predicted.	Not BTB-able instructions
Note: Other combinations reserved				

25.2.13.7 DPU Counter Triad B Control Register (DP_TBC)

DP_TBC DPU Triad B Control Register Offset 0x24

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		TDMP		TDM				—		TENMP		TENM			
Type	R/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
	—		CEGP		—			CEG					—	CMODE		TCEN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_TBC register is a 32-bit register that controls the operation of the DPU counter triad B, including what events and when they are counted. If the TCEN bit in the DP_TBC register is set, the appropriate counters are controlled by this register and ignore the programming of their own control register. When the TCEN bit is cleared, each counter is controlled individually by its own control register. **Table 25-20** defines the DP_TBC bit fields.

Table 25-19. DP_TBC Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
TDMP 29–28	0	**Triad Disable Mode Privilege Level** The event disabling the counters belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TDM 27–24	0	**Triad Disable Mode** The event that disables the counters in the triad.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000–1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	

Table 25-19. DP_TBC Bit Descriptions (Continued)

Name	Reset	Description	Settings
TENMP 21–20	0	**Triad Enable Mode Privilege Level** The event enabling the counters belongs to the task described by these bits. If the MARK instruction enables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
TENM 19–16	0	**Triad Enable Mode** The event that enables the counters.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
CEGP 13–12	0	**Counted Event Group Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CEG 8–4	0	**Counted Event Group** The source counted by the counter.	See **Table 25-18** for details.
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Triad Counters Mode** Specifies the mode of the counter	00 One shot. Each counters in the triad generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. Each counter in the triad has its value saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.

Table 25-19. DP_TBC Bit Descriptions (Continued)

Name	Reset	Description	Settings
TCEN 0	0	**Triad Control Enable** Determines whether the three counters in the triad are controlled by this control register or their individual control counters.	0 Each counter is controlled independently. 1 The three counters are controlled by this register and the individual register settings have no effect.

25.2.13.8 DPU Counter A0 Control Register (DP_CA0C)

DP_CA0C DPU Counter A0 Control Register Offset 0x2C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		CDMP		CDM				—		CENMP		CENM			
Type	R/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
	—		CEP		—			CE					—	CMODE		—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CA0C is a 32-bit register that controls the operation of the DPU Extension Support Counter A0, including what events and when they are counted. **Table 25-20** defines the DP_CA0C bit fields.

Table 25-20. DP_CA0C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000–1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	

Table 25-20. DP_CA0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA0 of the OCE (CEP bits can be 00 or 01) 00011 Number of interrupts 00100 Number of ICache thrashes due to miss. 00101– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	

Table 25-20. DP_CA0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.13.9 DPU Counter A0 Value Register (DP_CA0V)

DP_CA0V DPU Counter A0 Value Registers Offset 0x30

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

DP_CA0V is a 32-bit register containing the specified counter value. **Table 25-21** defines the counter bit fields.

Table 25-21. DP_CA0V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.13.10 DPU Counter A1 Control Register (DP_CA1C)

DP_CA1C DPU Counter A1 Control Register Offset 0x34

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		CDMP		CDM				—		CENMP		CENM			
Type	R/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
	—		CEP		—			CE					—	CMODE		—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CA1C is a 32-bit register that controls the operation of the DPU Extension Support Counter A1, including what events and when they are counted. **Table 25-22** defines the DP_CA1C bit fields.

Table 25-22. DP_CA1C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000–1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].

Table 25-22. DP_CA1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA1 of the OCE (CEP bits can be 00 or 01) 00011 Number of rollovers by counter A2 (CEP bits must be 00). 00100 Number of DCache thrashes due to miss. 00101– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.

Table 25-22. DP_CA1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.13.11 DPU Counter A1 Value Registers (DP_CA1V)

DP_CA1V DPU Counter A1 Value Registers Offset 0x38

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

DP_CA1V is a 32-bit register containing the specified counter value. **Table 25-21** defines the counter bit fields.

Table 25-23. DP_CA1V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.13.12 DPU Counter A2 Control Register (DP_CA2C)

DP_CA2C DPU Counter A2 Control Register Offset 0x3C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		CDMP		CDM				—		CENMP		CENM			
Type	R/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
	—		CEP		—			CE					—	CMODE		—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CA2C is a 32-bit register that controls the operation of the DPU Extension Support Counter A2, including what events and when they are counted. **Table 25-24** defines the DP_CA2C bit fields.

Table 25-24. DP_CA2C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-24. DP_CA2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA2 of the OCE (CEP bits can be 00 or 01) 00011 Number of task switches; includes the number of times the service of a task started including the first time (CEP bits must be 00). 00100– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10 Extension mode. The counter rolls over when it reaches 0 and continues to count. Counter A1 can be programmed to count the number of overflows. In this case, counter A1 must operate in one-shot mode. 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.13.13 DPU Counter A2 Value Registers (DP_CA2V)

DP_CA2V DPU Counter A2 Value Registers Offset 0x40

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

DP_CA2V is a 32-bit register containing the specified counter value. **Table 25-21** defines the counter bit fields.

Table 25-25. DP_CA2V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.13.14 DPU Counter B0 Control Register (DP_CB0C)

DP_CB0C DPU Counter B0 Control Register Offset 0x54

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		CDMP		CDM				—		CENMP		CENM			
Type	R/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
	—		CEP		—			CE					—	CMODE		—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CB0C is a 32-bit register that controls the operation of the DPU Extension Support Counter B0, including what events and when they are counted. **Table 25-28** defines the DP_CB0C bit fields.

Table 25-26. DP_CB0C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-26. DP_CB0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-26. DP_CB0C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA3 of the OCE (CEP bits can be 00 or 01) 00011 Number of interrupts 00100 Number of ICache thrashes due to miss. 00101– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.13.15 DPU Counter B0 Value Registers (DP_CB0V)

DP_CB0V DPU Counter B0 Value Registers Offset 0x58

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

DP_CB0V is a 32-bit register containing the specified counter value. **Table 25-21** defines the counter bit fields.

Table 25-27. DP_CB0V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.13.16 DPU Counter B1 Control Register (DP_CB1C)

DP_CB1C DPU Counter B1 Control Register Offset 0x5C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		CDMP		CDM				—		CENMP		CENM			
Type	R/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
	—		CEP		—			CE					—	CMODE		—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CB1C is a 32-bit register that controls the operation of the DPU Extension Support Counter B1, including what events and when they are counted. **Table 25-28** defines the DP_CB1C bit fields.

Table 25-28. DP_CB1C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-28. DP_CB1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-28. DP_CB1C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA4 of the OCE (CEP bits can be 00 or 01) 00011 Number of rollovers by counter B2 (CEP bits must be 00). 00100 Number of DCache thrashes due to miss. 00101– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10– 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.13.17 DPU Counter B1 Value Registers (DP_CB1V)

DP_CB1V DPU Counter B1 Value Registers Offset 0x60

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

DP_CB1V is a 32-bit register containing the specified counter value. **Table 25-29** defines the counter bit fields.

Table 25-29. DP_CB1V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.13.18 DPU Counter B2 Control Register (DP_CB2C)

DP_CB2C DPU Counter B2 Control Register Offset 0x64

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—		CDMP		CDM				—		CENMP		CENM			
Type	R/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
	—		CEP		—			CE					—	CMODE		—
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_CB2C is a 32-bit register that controls the operation of the DPU Extension Support Counter B2, including what events and when they are counted. **Table 25-30** defines the DP_CB2C bit fields.

Table 25-30. DP_CB2C Bit Descriptions

Name	Reset	Description	Settings
— 31–30	0	Reserved. Write to zero for future compatibility.	

Table 25-30. DP_CB2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CDMP 29–28	0	**Counter Disable Mode Privilege Level** The event disabling the counter belongs to the task described by these bits. If the DEBUGEV instruction disables the counters, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The disabling event belongs to any task. 01 The disabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The disabling event belongs to a supervisor level task. 11 The disabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CDM 27–24	0	**Counter Disable Mode** The event that disables the counter.	0000 No disabling event. 0001 DEBUGEV instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1111 reserved
— 23–22	0	Reserved. Write to zero for future compatibility.	
CENMP 21–20	0	**Counter Enable Mode Privilege Level** The event enabling the counter belongs to the task described by these bits. If the MARK instruction enables the counter, all the programming options mentioned here can be chosen. For EDCA events the privilege level can be filtered inside the EDCA itself.	00 The enabling event belongs to any task. 01 The enabling event is the result of a user task detected under the control of DP_CR[TIDCM]. 10 The enabling event belongs to a supervisor level task. 11 The enabling event is the result of a supervisor level task detected under the control of DP_CR[TIDCM].
CENM 19–16	0	**Counter Enable Mode** The event that enables the counter.	0000 The counter is disabled. 0001 MARK instruction. 0010 Event generated by EDCA0 in the OCE. 0011 Event generated by EDCA1 in the OCE. 0100 Event generated by EDCA2 in the OCE. 0101 Event generated by EDCA3 in the OCE. 0110 Event generated by EDCA4 in the OCE. 0111 Event generated by EDCA5 in the OCE. 1000– 1110 reserved 1111 The counter is enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	

Table 25-30. DP_CB2C Bit Descriptions (Continued)

Name	Reset	Description	Settings
CEP 13–12	0	**Counted Event Privilege Level** The source counted by the counter belongs to the task described by these bits.	00 The counter counts events that belong to any task. 01 The counter only counts events belonging to user tasks detected under the control of DP_CR[TIDCM]. 10 The counter counts event that belong to a supervisor level task. 11 The counter counts events belonging to supervisor level task detected under the control of DP_CR[TIDCM].
— 11–9	0	Reserved. Write to zero for future compatibility.	
CE 8–4	0	**Counted Event** The source counted by the counter.	00000 Clock cycles (non-debug) 00001 Application cycles (non-wait, non-stop, non-debug 00010 Number of events generated by the EDCA5 of the OCE (CEP bits can be 00 or 01) 00011 Number of task switches; includes the number of times the service of a task started including the first time (CEP bits must be 00). 00100– 11111 reserved
— 3	0	Reserved. Write to zero for future compatibility.	
CMODE 2–1	0	**Counter Mode** Specifies the mode of the counter	00 One shot. The counter generates an event when it reaches 0, stops counting, and disables itself. 01 Trace mode. The counter value is saved in a shadow register whenever required by the trace buffer. The counter continues to count and generates an event when it reaches 0. 10 Extension mode. The counter rolls over when it reaches 0 and continues to count. Counter B1 can be programmed to count the number of overflows. In this case, counter B1 must operate in one-shot mode. 11 reserved.
— 0	0	Reserved. Write to zero for future compatibility.	

25.2.13.19 DPU Counter B2 Value Registers (DP_CB2V)

DP_CB2V DPU Counter B2 Value Registers Offset 0x68

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

DP_CB2V is a 32-bit registers containing the specified counter value. **Table 25-31** defines the counter bit fields.

Table 25-31. DP_CA[0–2]V and DP_CB[0–2]V Bit Descriptions

Name	Reset	Description	Settings
— 31	0	Reserved. Write to zero for future compatibility.	
CV 30–0	0	**Counter Value** Stores the value of the counter.	

25.2.13.20 DPU Trace Control Register (DP_TC)

DP_TC DPU Trace Control Register Offset 0x7C

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	—					SHARE	—	CSS		—	PRIV	GLOBAL			BURST_SIZE	
Type	R/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
	—			TMPDIS	—		VTBWM		—	SAMPLE	—	TMODE				EN
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DP_TC is a 32-bit register responsible that controls the VTB memory. **Table 25-32** defines the DP_TC bit fields.

Table 25-32. DP_TC Bit Descriptions

Name	Reset	Description	Settings
— 31–27	0	Reserved. Write to zero for future compatibility.	

Table 25-32. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
SHARE 26	0	**Share** Used when the DSP core subsystem operates in mixed endian mode. This bit allows the VTB to operate in a memory range shared with a system using little-endian mode.	0 Cannot share with system using little-endian mode. 1 Can share with system using little-endian mode.
— 25	0	Reserved. Write to zero for future compatibility.	
CSS 24–23	0	**Chip Select** The chip select attribute of the VTB write access.	00 Reserved 01 Reserved 10 L2 Cache 11 Reserved
— 22	0	Reserved. Write to zero for future compatibility.	
PRIV 21	0	**Privilege Mode** The privilege attribute of the VTB write access.	0 User 1 Supervisor
GLOBAL 20–18	0	**Global Attribute** The global attribute of the VTB write access	000 Cacheable write-through 001 Cacheable write-back 010 Non-cacheable write-through 011–111 reserved
BURST_ SIZE 17–16	0	**Burst Size** The burst size of the VTB write access.	00 1 VBR (16 bytes) 01 2 reserved 10 4 VBRs (64 bytes) 11 8 reserved
— 15–13	0	Reserved. Write to zero for future compatibility.	

Table 25-32. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
TMPDIS 12	0	**Temporary Disable** To verify that all traced information arrived at the VTB, a flush can be generated that is similar to the flush generated when tracing is disabled. The only difference is that when tracing is disabled by setting the TMPDIS bit, when tracing is re -enabled by resetting this bit, the value of the Start Address is not sampled at the TB Write Pointer. The TB Write Pointer saves its previous value, thereby enabling tracing to be continued from the point where it was interrupted. If the tracing is enabled, TMPDIS bit must not be changed in debug mode by Core Command. The TMPDIS bit must not be set together with the EN bit. When the tracing is disabled by resetting the EN bit, the TMPDIS bit should not be set. When the TMPDIS bit is set, its resetting should not be done in the 16 VLES after the setting. Before the tracing is disabled by setting TMPDIS bit, the user has to wait until the TWB finishes the previous flush that can be initiated by a previous trace disabling or by entering wait or stop states. (Bit TWBA in the DP_SR register indicates if the TWB is in the middle of a flush operation.) In order not to lose information, serving an interrupt between the checking of the TWBA bit and the disabling should be avoided. (It can be done by disabling the interrupts before the checking of the TWBA bit and enabling them after the disabling.) When the TMPDIS bit is set, the trace logic disregards further inputs.	0 Trace logic enabled 1 Trace logic disabled
— 11–10	0	Reserved. Write to zero for future compatibility.	
VTBWM 9–8	0	**Virtual Trace Buffer Write Mode** Specifies the manner in which the VTB is written.	00 Overwrite mode. The DPU writes all the time. After writing to the End Address, the write pointer wraps to the first address and starts to overwrite the first entries. When disabled, it contains the last the entries leading to the disable point. (The TBF bit in the DP_MR register shows that the VTB was full at least once.) 01 One address mode. The DPU always writes to the same address (described by the Start Address). 10 Trace event request mode. When the write access is generated to the address equal to the value in the Trace Event Request, then depending on the programming of the DETB bit in the DP_CR register, either an interrupt to the EPIC or a debug request to the core (to the OCE) is generated. The debug request to the core may become an internal debug exception according to the programming of the OCE. 11 reserved
— 7	0	Reserved. Write to zero for future compatibility.	

Table 25-32. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
SAMPLE 6	0	**Sample** A bit that can only be set. When set, the counter values are sampled to the trace buffer. This bit is only relevant when the TMODE bits are 0100. This bit is cleared by the DPU after writing the information to the trace buffer.	0 No action 1 Counter values are added to the trace buffer.
— 5	0	Reserved. Write to zero for future compatibility.	
TMODE 4–1	0	**Trace Mode** Identifies the source that is written to the trace buffer.	0000 Information generated by the OCE. Data compression is off. 0001 Information generated by the OCE after compression and adding task ID flags. 0010 Depending on the OCE programming, the data transferred at a task switch is either the first and last PC of the tasks together with a task flag and all six counter values; or, for a jump to a subroutine, a jump to an interrupt routine, or a return from either, the task ID and the values of all six counters. 0011 reserved 0100 When the SAMPLE bit in the Trace Buffer Control register is set by software at specified points during the application execution, the task ID and all six counters. 1010 The value written to the Trace Buffer Data register every time a new value is written. 0110 The task ID and the value of all six counters when EDCA5 generates an event that belongs to the task described by the task ID comparator. When the task is a user task, the task ID comparator must be programmed accordingly by the TIDCM bits in the DP_CR register. When the task is a supervisor level task or any task, the TIDCM bits in the DP_CR register must be 00. 0111– 1000 reserved

Table 25-32. DP_TC Bit Descriptions (Continued)

Name	Reset	Description	Settings
EN 0	0	**Enable** This bit is used to enable/disable the trace buffer. Use the following guidelines to enable/disable the trace buffer: • Always wait until any current flush operations are completed before disabling a trace operation. Poll the DP_SR[TWBA] bit to determine the flush status. • To prevent any interrupt servicing that may occur between reading the DP_SR[TWBA] bit and disabling the trace buffer, always disable the interrupts before reading the DP_SR[TWBA] bit and enable them only after disabling the trace buffer. • Never change the configuration of a trace operation during execution. Always disable the trace buffer first and then change the configuration.	0 Trace buffer disabled. 1 Trace buffer enabled

25.2.13.21 DPU VTB Start Address Register (DP_TSA)

DP_TSA DPU VTB Start Address Register Offset 0x80

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

The DP_TSA is a 32-bit register that contains the start address of the VTB (physical address). The VTB can be located only in Bank 3 (between addresses 0x00000000–0xFF000000). Alignment depends on the burst size:

- For a burst size of 1 VBR, TER must be aligned to the value: 32 × Burst Size (programmed in the DP_TC register).
- For a burst size of 4 VBRs, the value of TER can be 32 × (2 × n – 1), where n is a positive integer.

Note: Bits 4–0 must be written as zeros and are read as zeros.

Table 25-34 defines the DP_TSA bit fields.

Table 25-33. DP_TSA Bit Descriptions

Name	Reset	Description	Settings
SA 31–5	0	**Start Address** Specifies the end address of the VTB address range.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.13.22 DPU VTB End Address Register (DP_TEA)

DP_TEA DPU VTB End Address Register Offset 0x84

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

The DP_TEA is a 32-bit register that contains the end address of the VTB (physical address). The VTB can be located only in Bank 3 (between addresses 0x00000000–0xFF000000). Alignment depends on the burst size:

- For a burst size of 1 VBR, TER must be aligned to the value: 32 × Burst Size (programmed in the DP_TC register).
- For a burst size of 4 VBRs, the value of TER can be $32 \times (2 \times n - 1)$, where n is a positive integer.

Note: Bits 4–0 must be written as zeros and are read as zeros.

Table 25-34 defines the DP_TEA bit fields.

Table 25-34. DP_TEA Bit Descriptions

Name	Reset	Description	Settings
EA 31–5	0	**End Address** Specifies the end address of the VTB address range.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.13.23 DPU Trace Event Request Register (DP_TER)

DP_TER DPU Trace Event Request Register Offset 0x88

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

The DP_TER is a 32-bit register that is used when the VTB is written in Trace Event Request mode (programmed in the DP_TC[TMODE] field). The DP_TER register contains the address within the range of the VTB where an interrupt or debug request should be generated (depending on the programming of the DETB bit in the DP_CR register). Alignment depends on the burst size:

- For a burst size of 1 VBR, TER must be aligned to the value: 32 × Burst Size (programmed in the DP_TC register).
- For a burst size of 4 VBRs, the value of TER can be 32 × (2 × n – 1), where n is a positive integer.

Note: Bits 4–0 must be written as zeros and are read as zeros.

Table 25-37 defines the DP_TER bit fields.

Table 25-35. DP_TER Bit Descriptions

Name	Reset	Description	Settings
TER 31–5	0	**Trace Event Request** Specifies the address within the VTB address range where the interrupt or debug request is generated.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.13.24 DPU Trace Write Pointer Register (DP_TW)

DP_TW DPU Trace Write Pointer Register Offset 0x8C

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

The DP_TW is a 32-bit register that contains the pointer to the location in the VTB where the last entry was written by the DPU. When tracing is enabled (by setting the DP_TC[EN] bit), the value of the VTB Start Address register is sampled to it. When tracing is re-enabled (by resetting the DP_TC[TMPDIS] bit), the value of the TB Write Pointer does not change. If it is written by the user, its value must be aligned to the value “32 × Burst Size” programmed in the DP_TC register.

Note: Before changing the value of the DP_TW register, you must generate a flush by setting the DP_TC[TMPDIS] bit.

Table 25-37 defines the DP_TW bit fields.

Table 25-36. DP_TW Bit Descriptions

Name	Reset	Description	Settings
WP 31–5	0	**Write Pointer** Stores the pointer to the last entry written by the DPU to the VTB.	
— **4–0**	0	Reserved. Write to zero for future compatibility.	

25.2.13.25 DPU Trace Data Register (DP_TD)

DP_TD DPU Trace Data Register Offset 0x90

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

The DP_TD is a 32-bit register containing the value to be written to the TB. This value is only relevant when the MODE bits in the DP_TC register are 1010.

Table 25-37 defines the DP_TD bit fields.

Table 25-37. DP_TD Bit Descriptions

Name	Reset	Description	Settings
TBD 31–0	0	**Trace Buffer Data** Stores the data to write to the Trace Buffer.	

25.3 Performance Monitor

The MSC8156E includes a performance monitor facility that can be used to monitor and record selected behaviors of the integrated device. **Section 25.3.1.5** briefly describes the events that can be monitored. Refer to the individual module chapters for a better understanding of these events. Performance monitor up-counters (PMC0–PMC8) count events selected by the performance monitor local control registers. PMC0 is a 64-bit up-counter specifically designated to count cycles. PMC1–PMC8 are 32-bit counters that can monitor 64 specific events in addition to counting 64 reference events. Each counter is associated with two local control registers (A and B) that configure the events counted. In addition, there is a global control register that can be used to enable/disable all the counters at one time.

The benefits of an internal performance monitor are numerous, and include the following:

- Because some systems or software environments are not easily characterized by signal traces or benchmarks, the performance monitor can be used to understand the MSC8156E device behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.

- System performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

Figure 25-15 is a high-level block diagram of the performance monitor. The module consists of a global control register (PMGC), one 64-bit counter (PMC0), eight 32-bit counters, and two control registers per counter. The global control register PMGC affects all counters and takes priority over local control registers. The local control registers are divided into two groups, as follows:

- Local control A registers control counter freezing, overflow condition enable, and event selection. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers contain the counters' threshold values. Local control register PMLCB0 is not used.

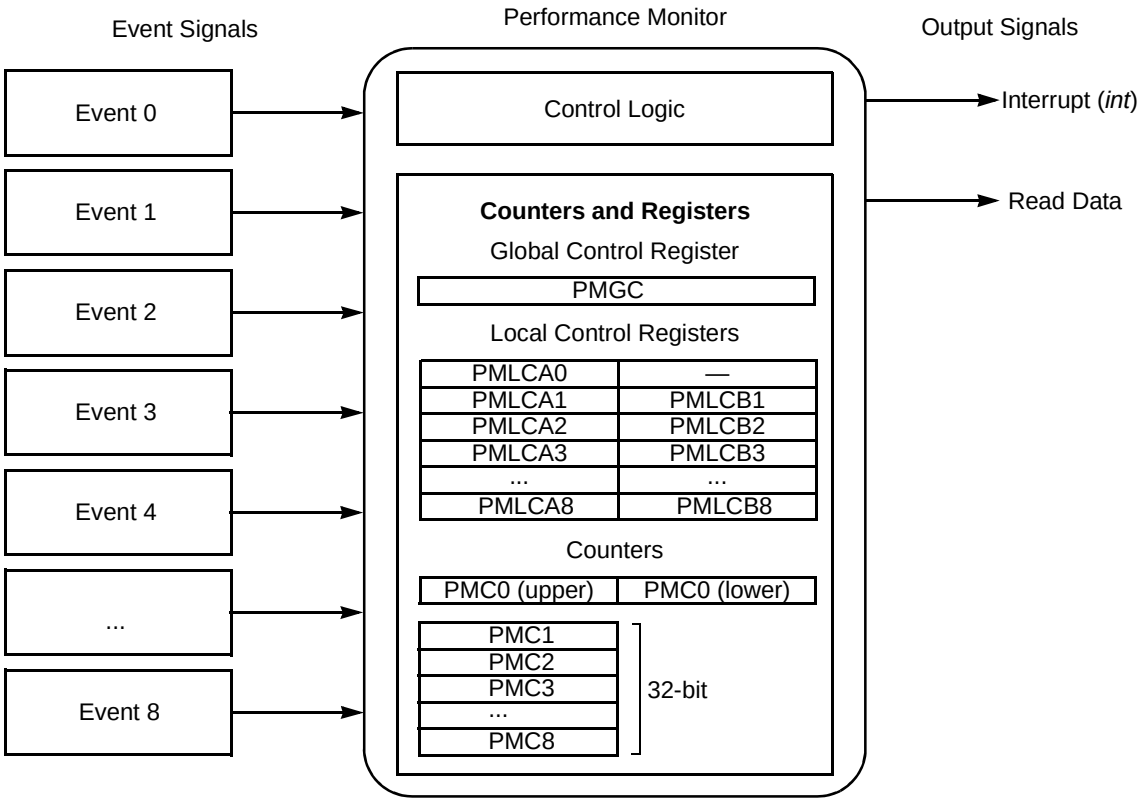


Figure 25-15. Performance Monitor Block Diagram

Performance monitor events are signalled by the functional blocks in the integrated device and are selectively recorded in the PMCs. Sixty-four of these events are referred to as reference events, which can be counted on any of the eight counters. Counter-specific events can be counted only on the counter for which the event is defined.

The performance monitor can generate an interrupt on overflow. Several control registers specify how a performance monitor interrupt is signalled. The PMCs can also be programmed to freeze when an interrupt is generated.

25.3.1 Functional Description

The MSC8156E performance monitor offers a rich set of features that permits a complete performance characterization of the implementation. These features include:

- One 64-bit counter exclusively dedicated to counting cycles.
- Eight 32-bit counters that count the occurrence of selected events.
- One global control register (all counters) and two local control registers per counter.
- Counts up to 64 reference events on any of the eight 32-bit counters.
- Ability to count up to 512 counter-specific events.
- chaining capability.
- Quantity threshold counting.
- Ability to generate an interrupt on overflow.

The performance monitor does not drive any signals externally, but it does assert the internal interrupt signal when a monitored event or other interrupt condition occurs.

25.3.1.1 Performance Monitor Interrupts

The PMCs can generate an interrupt on an overflow when the msb of a counter changes from 0 to 1. For the interrupt to be signalled, the condition enable bit (PMLCA*n*[CE]) and performance monitor interrupt enable bit (PMGC[PMIE]) must be set. When an interrupt is signalled and the freeze-counters-on-enabled-condition-or-event bit (PMGC[FCECE]) is set, PMGC[FAC] is set by hardware and all of the registers are frozen. Software can clear the interrupt condition by resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

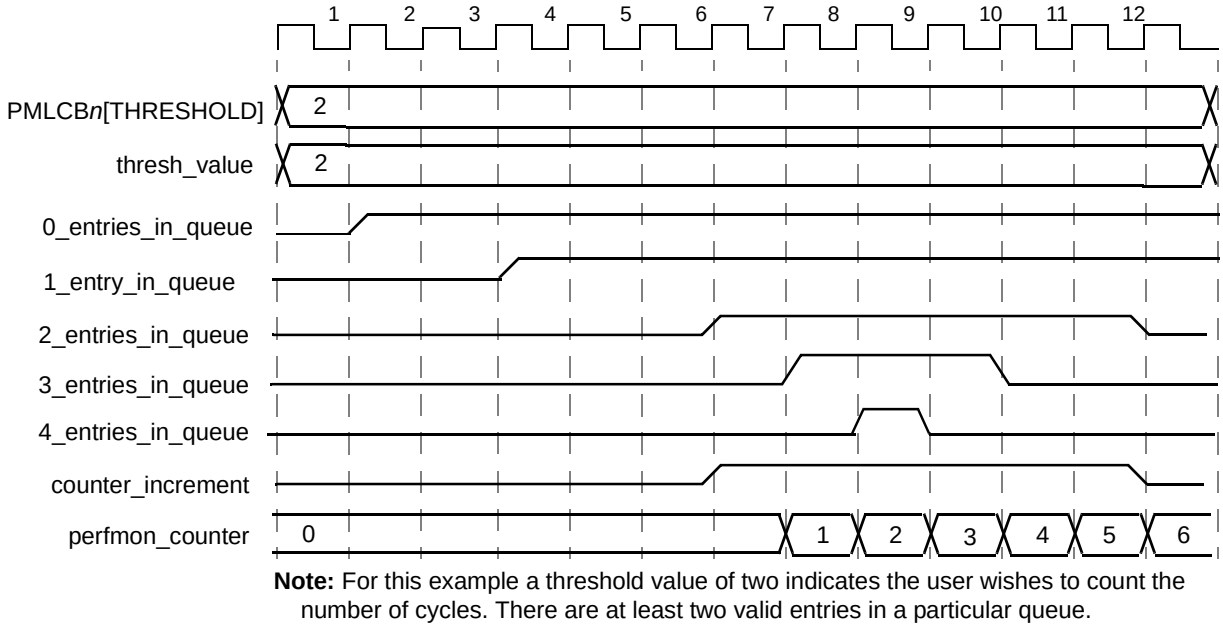
25.3.1.2 Event Counting

Using the control registers described in **Section 25.3.2**, the nine PMCs can count the occurrences of specific events. The 64-bit PMC0 is design to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC8). Additionally, up to 64 unique events can be counted on each 32-bit counter. The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.

Note: Using PMLCAn[FC] resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

25.3.1.3 Threshold Events

The quantity threshold event sequences the performance monitor counter is only incremented when the specified threshold event exceeds the threshold value. Threshold event is generally used to monitor the usage of buffers and queues. For example, the usage of a specific queue can be characterized by measuring the amount of time the queue is completely full or partially full. For this example the threshold field would be used to specify how many entries are required to be valid in the queue for that event to be counted.A timing diagram for quantity threshold event counting is shown in **Figure 25-16**.



Note: For this example a threshold value of two indicates the user wishes to count the number of cycles. There are at least two valid entries in a particular queue.

Figure 25-16. Quantity Threshold Event Sequence Timing Diagram

25.3.1.4 Chaining

By configuring one counter to increment each time another counter overflows, several counters can be chained together to provide event counts larger than 32 bits. Each counter in a chain adds 32 bits to the maximum count. The register chaining sequence is not arbitrary and is specified indirectly by selecting the register overflow event to be counted. Selecting an event has the effect of selecting a source register because all available chaining events, as shown in **Table 25-38**, are dedicated to specific registers.

Note: The chaining overflow event occurs when the counter reaches its maximum value and wraps, not when the register msb is set. For this overflow to occur, PMLCA*n*[CE] should be cleared to avoid signalling an interrupt when the counter most significant bit

is set. Several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

25.3.1.5 Performance Monitor Events

Table 25-38 lists performance monitor events specified in PMLCA[1–8]. The event assignment column indicates the event type and number, using the following formats:

- Ref:#—Reference events are shared across counters PMC1–PMC8. The number indicates the event. For example, Ref:6 means that PMC1–PMC8 share reference event 6.
- C[0–8]:#—Counter-specific events. C8 indicates an event assigned to PMC8. Thus C8:62 means PMC8 is assigned event 62 (RapidIO DMA1 Channel 2 Write DW).

Note: With counter-specific events, an offset of 64 must be used when programming the field, because counter-specific events occupy the bottom 64 values of the 7-bit event field where events are numbered. For example, to specify counter-specific event 0, the event field must be programmed to 64.

Counter events not specified in **Table 25-38** are reserved.

Note: The DMA event frequency is different from the PM frequency. Therefore, these events pass through a synchronizer before entering the PM.

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
General Events		
Nothing	Ref:0	Register counter holds current value
System cycles	C0	CCB HSSI clock cycles
System DMA Events		

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Asserted for every cycle DMA specific channel is active	C5:3	This event counts the number of cycles a specific DMA channel is active. It is insufficient to enable the C5:3 event. Three additional parameters must be defined in the DMA: • en_profiling. Enables/disables DMA profiling. – 0 = Disable DMA profiling – 1 = Enable DMA profiling • channel_number. Selects the DMA channel to profile. – 000000–001111 = Channel number – 01xxxx = Reserved – 10xxxx = Reserved • dest_ch_profiler. Specifies whether to profile the source or destination requests. – 0 = Source – 1 = Destination For detaila about DMA profiling, see **Section 14.5**, *Profiling*, on page 14-23.
RapidIO RMU EVENTS		
Packet received of any priority	C1:5	
Packet received of priority 0	C2:5	
Packet received of priority 1	C3:5	
Packet received of priority 2	C4:5	
Packet received of priority 3	C5:5	
Clock cycle occurred in which the inbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.	C1:46	
Clock cycle occurred in which the inbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.	C2:46	
Clock cycle occurred in which the inbound buffer is full to priority 1. Event asserted for as many clock cycles as this is true.	C3:46	
Clock cycle occurred in which the inbound buffer is full to priority 2. Event asserted for as many clock cycles as this is true.	C4:46	
Clock cycle occurred in which the inbound buffer is full to priority 3. Event asserted for as many clock cycles as this is true.	C5:46	

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
SRIO0 Events		
A received packet has been sent to OCN for passthrough	C2:2	
Packet sent to RapidIO	C4:2	
Packet was retried	C6:2	
Packet was reordered	C8:2	Packet was reordered on Outbound. After an outbound request is retried, the Serial RapidIO controller preferentially tries to send the oldest next higher priority packet.
Packet sent to RapidIO of priority 0	C5:8	
Packet sent to RapidIO of priority 1	C6:8	
Packet sent to RapidIO of priority 2	C7:8	
Packet sent to RapidIO of priority 3	C8:8	
Clock cycle occurred in which the outbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.	C4:15	Outbound logical buffer in the Serial RapidIO controller.
Clock cycle occurred in which the outbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.	C5:15	
Clock cycle occurred in which the outbound buffer is full to priority 1. Event asserted for as many clock cycles as this is true.	C6:15	
Clock cycle occurred in which the outbound buffer is full to priority 2. Event asserted for as many clock cycles as this is true.	C7:15	
Clock cycle occurred in which the outbound buffer is full to priority 3. Event asserted for as many clock cycles as this is true.	C8:15	
Packet accepted on RapidIO	C5:2	
Packet accepted from RapidIO of priority 0	C6:9	
Packet accepted from RapidIO of priority 1	C7:9	
Packet accepted from RapidIO of priority 2	C8:9	
Packet accepted from RapidIO of priority 3	C1:9	
Packet retry occurred due to inbound buffer limitations for any priority	C7:2	
Packet retry occurred due to inbound buffer limitations, priority 0	C8:60	
Packet retry occurred due to inbound buffer limitations, priority 1	C1:60	
Packet retry occurred due to inbound buffer limitations, priority 2	C2:60	
Packet retry occurred due to inbound buffer limitations, priority 3	C3:60	
Clock cycle occurred in which the inbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.	C2:13	
Clock cycle occurred in which the inbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.	C3:13	

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Clock cycle occurred in which the inbound buffer is full to priority 1. Event asserted for as many clock cycles as this is true.	C4:13	
Clock cycle occurred in which the inbound buffer is full to priority 2. Event asserted for as many clock cycles as this is true.	C5:13	
Clock cycle occurred in which the inbound buffer is full to priority 3. Event asserted for as many clock cycles as this is true.	C6:13	
RapidIO DMA0 Events		
DMA Channel 0 Read request	C1:0	OCNDMA data read only.
DMA Channel 1 Read request	C2:0	OCNDMA data read only.
DMA Channel 2 Read request	C3:0	OCNDMA data read only.
DMA Channel 3 Read request	C4:0	OCNDMA data read only.
DMA Channel 0 Write request	C6:53	OCNDMA write.
DMA Channel 1 Write request	C7:53	OCNDMA write.
DMA Channel 2 Write request	C8:53	OCNDMA write.
DMA Channel 3 Write request	C5:53	OCNDMA write.
DMA Channel 0 Descriptor request	C7:54	
DMA Channel 1 Descriptor request	C8:54	
DMA Channel 2 Descriptor request	C5:54	
DMA Channel 3 Descriptor request	C6:54	
Channel 0 Read DW	C8:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 1 Read DW	C5:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 2 Read DW	C6:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 3 Read DW	C7:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 0 Write DW	C2:62	OCNDMA descriptor write DW = 8 bytes
Channel 1 Write DW	C3:62	OCNDMA descriptor write DW = 8 bytes
Channel 2 Write DW	C4:62	OCNDMA descriptor write DW = 8 bytes
Channel 3 Write DW	C1:62	OCNDMA descriptor write DW = 8 bytes
SRIO1 Events		
A received packet has been sent to OCN for passthrough	C3:1	
Packet sent to RapidIO	C7:1	
Packet was retried	C4:1	

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Packet was reordered	C1:1	Packet was reordered Outbound. After an outbound request gets retried, the Serial RapidIO controller preferentially tries to send the oldest next higher priority packet.
Packet sent to RapidIO of priority 0	C8:11	
Packet sent to RapidIO of priority 1	C1:11	
Packet sent to RapidIO of priority 2	C2:11	
Packet sent to RapidIO of priority 3	C3:11	
Clock cycle occurred in which the outbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.	C5:16	Outbound logical buffer in the Serial RapidIO controller.
Clock cycle occurred in which the outbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.	C6:16	
Clock cycle occurred in which the outbound buffer is full to priority 1. Event asserted for as many clock cycles as this is true.	C7:16	
Clock cycle occurred in which the outbound buffer is full to priority 2. Event asserted for as many clock cycles as this is true.	C8:16	
Clock cycle occurred in which the outbound buffer is full to priority 3. Event asserted for as many clock cycles as this is true.	C1:16	
Packet accepted on RapidIO	C8:1	
Packet accepted from RapidIO of priority 0	C1:12	
Packet accepted from RapidIO of priority 1	C2:12	
Packet accepted from RapidIO of priority 2	C3:12	
Packet accepted from RapidIO of priority 3	C4:12	
Packet retry occurred due to inbound buffer limitations for any priority	C5:1	
Packet retry occurred due to inbound buffer limitations, priority 0	C6:59	
Packet retry occurred due to inbound buffer limitations, priority 1	C7:59	
Packet retry occurred due to inbound buffer limitations, priority 2	C8:59	
Packet retry occurred due to inbound buffer limitations, priority 3	C1:59	
Clock cycle occurred in which the inbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.	C3:14	
Clock cycle occurred in which the inbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.	C4:14	
Clock cycle occurred in which the inbound buffer is full to priority 1. Event asserted for as many clock cycles as this is true.	C5:14	
Clock cycle occurred in which the inbound buffer is full to priority 2. Event asserted for as many clock cycles as this is true.	C6:14	

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
Clock cycle occurred in which the inbound buffer is full to priority 3. Event asserted for as many clock cycles as this is true.	C7:14	
RapidIO DMA1 Events		
DMA Channel 0 Read request	C5:0	OCNDMA data read only.
DMA Channel 1 Read request	C6:0	OCNDMA data read only.
DMA Channel 2 Read request	C7:0	OCNDMA data read only.
DMA Channel 3 Read request	C8:0	OCNDMA data read only.
DMA Channel 0 Write request	C2:53	OCNDMA write.
DMA Channel 1 Write request	C3:53	OCNDMA write.
DMA Channel 2 Write request	C4:53	OCNDMA write.
DMA Channel 3 Write request	C1:53	OCNDMA write.
DMA Channel 0 descriptor request	C3:54	
DMA Channel 1 descriptor request	C4:54	
DMA Channel 2 descriptor request	C1:54	
DMA Channel 3 descriptor request	C2:54	
Channel 0 Read DW	C4:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 1 Read DW	C1:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 2 Read DW	C2:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 3 Read DW	C3:61	OCNDMA descriptor read only. DW = 8 bytes
Channel 0 Write DW	C6:62	OCNDMA descriptor write DW = 8 bytes
Channel 1 Write DW	C7:62	OCNDMA descriptor write DW = 8 bytes
Channel 2 Write DW	C8:62	OCNDMA descriptor write DW = 8 bytes
Channel 3 Write DW	C5:62	OCNDMA descriptor write DW = 8 bytes
PCI Express Events		
PM inbound OCN read request. Indicates that the bridge has request an OCN read.	C1:32	
PM inbound OCN write request. Indicates that the bridge has request an OCN write.	C2:32	
PM inbound OCN data valid. Indicates that the bridge has valid data to send. This is true for write.	C5:30	
PM outbound OCN read request. When asserted indicates that the bridge has received an OCN read packet.	C8:32	

Table 25-38. Performance Monitor Events Performance

Event Counted	Number	Description of Event Counted
PM outbound OCN write request. When asserted indicates that the bridge has received an OCN write packet.	C3:31	
PM outbound OCN data valid. When asserted indicates that the bridge has received OCN write data.	C4:31	
Chaining Events		
PMC0 carry-out	Ref:1	PMC0[63] 1-to-0 transitions.
PMC1 carry-out	Ref:2	PMC1[63] 1-to-0 transitions. Reserved for PMC1.
PMC2 carry-out	Ref:3	PMC2[63] 1-to-0 transitions. Reserved for PMC2.
PMC3 carry-out	Ref:4	PMC3[63] 1-to-0 transitions. Reserved for PMC3.
PMC4 carry-out	Ref:5	PMC4[63] 1-to-0 transitions. Reserved for PMC4.
PMC5 carry-out	Ref:6	PMC5[63] 1-to-0 transitions. Reserved for PMC5.
PMC6 carry-out	Ref:7	PMC6[63] 1-to-0 transitions. Reserved for PMC6.
PMC7 carry-out	Ref:8	PMC7[63] 1-to-0 transitions. Reserved for PMC7.
PMC8 carry-out	Ref:9	PMC8[63] 1-to-0 transitions. Reserved for PMC8.

25.3.1.6 Performance Monitor Examples

Table 25-40 contains sample register settings for the three supported modes.

- Simple event performance monitoring example
- Threshold event performance monitoring example

The settings in **Table 25-39** are identical for all three examples.

Table 25-39. PMGC and PMLCA*n* Settings

Field	Setting	Reason
PMGC[FAC]	0	Counters must not be frozen.
PMGC[PMIE]	1	Performance monitor interrupts are enabled
PMGC[FCECE]	1	Counters should be frozen when an interrupt is signalled.
PMLCA*n*[FC]	0	Counters cannot be frozen for counting.
PMLCA*n*[CE]	1	Overflow condition enable is required to allow interrupt signalling.

For simple event counting, a non-threshold event is selected in PMLCA*n*[EVENT] and all other features are disabled by clearing all register fields except for CE.

For threshold counting, a threshold event must be specified in PMLCA*n*[EVENT] and threshold value in PMLCB*n*[THRESHOLD].

Table 25-40. Register Settings for Counting Examples

Register	Register Field	Simple Event	Threshold
PMGC	FAC	0	0
	PMIE	1	1
	FCECE	1	1
PMLCA*n*	FC	0	0
	CE	1	1
	EVENT	121	33
PMLCB*n*	TRIGONSEL	0	0
	TRIGOFFSEL	0	0
	TRIGONCNTL	0	0
	TRIGOFFCNTL	0	0
	THRESHOLD	0	3

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

25.3.2 Performance Monitor Programming Model

The performance monitor system includes the following registers:

- Performance Monitor Global Control Register (PMGC), see **page 25-80**.
- Performance Monitor Local Control Register A0 (PMLCA0), see **page 25-81**.
- Performance Monitor Local Control Register A[1–8] (PMLCA[1–8]), see **page 25-82**.
- Performance Monitor Local Control Register B[1–8] (PMLCB[1–8]), see **page 25-83**.
- Performance Monitor Counter 0 (PMC0), see **page 25-84**.
- Performance Monitor Counter 1–8 (PMC[1–8]), see **page 25-85**.

Note: Because accessing a PMC manually has priority over counter incrementation by the counted event, reading or writing a PMC while it is counting may affect the count. Likewise, accessing a performance monitor control register while its target counter is counting may also affect the count.

Note: The Performance Monitor block uses the base address: 0xFFFBB800.

25.3.2.1 Performance Monitor Global Control Register (PMGC)

PMGC Performance Monitor Global Control Register Offset 0x00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FAC	PMIE	FCECE	—												
Type	R/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**
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMGC globally configures the PMC operation. **Table 25-41** defines the PMGC bit fields.

Table 25-41. PMGC Bit Descriptions

Name	Reset	Description	Settings
FAC 31	0	**Freeze All Counters** Enables or freezes all counters. This bit is set by hardware when a performance monitor interrupt occurs and the FCECE bit is set.	0 PMCs increment if permitted by other control bits 1 PMCs do not increment.
PMIE 30	0	**Performance Monitor Interrupt Enable** Enables/disables the performance monitor interrupt. When enabled, the interrupt is asserted when a PMC overflows.	0 Interrupts disabled. 1 Interrupts enabled.
FCECE 29	0	**Freeze Counters on Enabled Condition or Event** Determines whether a PMC can continue increment if permitted by other control bits, or freezes when an enabled condition or event occurs.	0 PMCs increment. 1 PMCs freeze when the enabled condition or event occurs.
— 28–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.2 Performance Monitor Local Control A0 Register (PMLCA0)

PMLCA0 Performance Monitor Local Control A0 Offset 0x10

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

PMLCA0 configures the PMC0 operation. **Table 25-42** defines the PMLCA0 bit fields.

Table 25-42. PMLCA0 Bit Descriptions

Name	Reset	Description	Settings
FC 31	0	**Freeze Counter 0** Enables or freezes PMC0.	0 PMC0 increments if permitted by other control bits. 1 PMC0 disabled and does not increment.
— 30–27	0	Reserved. Write to zero for future compatibility.	
CE 26	0	**Condition Enable** Controls the counter 0 overflow condition. This bit should be cleared when PMC0 is selected for chaining. An overflow condition occurs when PMC0[msb] is set	0 Overflow cannot occur. PMC0 cannot cause interrupts or freeze counters. 1 Overflow conditions are enabled and can generate interrupts and freeze counters.
— 25–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.3 Performance Monitor Local Control A[1–8] (PMLCA[1–8])

PMLCA[1–8] Performance Monitor Local Control A[1–8] Offset 0x10 + n*0x10

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FC	—				CE	—			EVENT						
Type	R/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
	—															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMLCA[1–8], along with PMLCB[1–8], configure the respective PMC[1–8] operation. **Table 25-43** defines the PMLCA[1–8] bit fields.

Table 25-43. PMLCA[1–8] Bit Descriptions

Name	Reset	Description	Settings
FC 31	0	**Freeze Counter** Enables or freezes PMCn.	0 PMCn increments if permitted by other control bits. 1 PMCn disabled and does not increment.
— 30–27	0	Reserved. Write to zero for future compatibility.	
CE 26	0	**Condition Enable** Controls the counter n overflow condition. This bit should be cleared when PMCn is selected for chaining. **Note:** An overflow condition occurs when PMCn[msb] is set.	0 Overflow cannot occur. PMCn cannot cause interrupts or freeze counters. 1 Overflow conditions are enabled and can generate interrupts and freeze counters.
— 25–23	0	Reserved. Write to zero for future compatibility.	
EVENT 22–16	0	**Event Selector** Selects the event to count.	Bit 22 is always set (1). This is because of the required offset of 64. See the first note in **Section 25.3.1.5**. The definitions for Bits [21:16] values 0x00 to 0x3F are Event Numbers defined in **Table 25-38**.
— 15–0	0	Reserved. Write to zero for future compatibility.	

25.3.2.4 Performance Monitor Local Control B[1–8] (PMLCB[1–8])

PMLCB[1–8] Performance Monitor Local Control B[1–8] Offset 0x14 + n*0x10

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

PMLCB[1–8], along with PMLCA[1–8], configure the respective PMC[1–8] operation. **Table 25-44** defines the PMLCB[1–8] bit fields.

Table 25-44. PMLCB[1–8] Bit Descriptions

Name	Reset	Description	Settings
— 31–6	0	Reserved. Write to zero for future compatibility.	
THRESHOLD 5–0	0	**Threshold** Sets the counting threshold level. Only event with a number of occurrence greater than this number are counted.	

25.3.2.5 Performance Monitor Counter 0 (PMC0)

PMC0 Performance Monitor Counter 0 Offset 0x18

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
	PMC0															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
	PMC0															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PMC0															
Type	R/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
	PMC0															
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC0 is only used to count clock cycles for events specified by PMLCA0. **Table 25-45** defines the PMC0 bit fields.

Table 25-45. PMC0 Bit Descriptions

Name	Reset	Description	Settings
PMC0 63–0	0	**Performance Monitor Counter 0** Contains the current PMC0 count.	

25.3.2.6 Performance Monitor Counter 1–8 (PMC[1–8])

PMC[1–8] Performance Monitor Counter 1–8 Offset 0x18 + n*0x10

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

PMC[1–8] are used to count events selected by the corresponding performance monitor local control registers. **Table 25-46** defines the PMC[1–8] bit fields.

Table 25-46. PMC[1–8] Bit Descriptions

Name	Reset	Description	Settings
PMCn 31–0	0	**Performance Monitor Counter n** Contains the current PMCn count.	

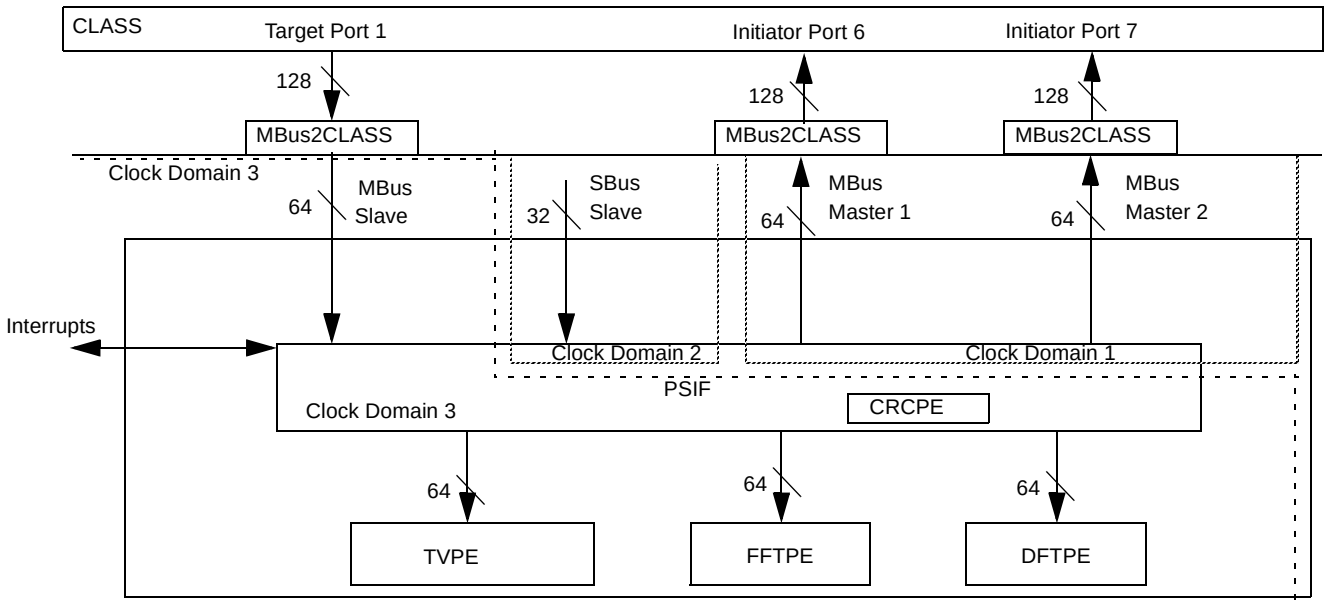
Multi Accelerator Platform Engine, Baseband (MAPLE-B)

26

The MAPLE-B is a baseband algorithm accelerator for Turbo, Viterbi, FFT/iFFT, DFT/iDFT, and CRC algorithms. The MAPLE-B consists of a Programmable-System-Interface (PSIF), which is a programmable controller with CRC accelerator and DMA capabilities, and three accelerators attached to it using an IP interface: the TVPE (Turbo/Viterbi Processing-Element), the FFTPE (FFT Processing-Element) and DFTPE (DFT Processing-Element). The PSIF has two 64-bit wide MBus master ports, used to transfer input and output data to and from system memory, and a 64-bit MBus Slave port, that allows any host to access its internal memories. The PSIF internal registers are accessed using the SBus.

Host(s) access internal memories to: perform the following functions:

- Place buffer-descriptors in the PSIF internal memory
- Access the MAPLE-B parameter RAM
- Access the Process Element (PE) registers.



Note: Because the MBus2CLASS bridges convert 64-bit data to 128-bit data, when using the CLASS profiler, every two sets of 64-bit data is converted to a single 128-bit data acknowledge.

Figure 26-1. MAPLE-B Overview

26.1 Features

The MAPLE-B supports the following:

- Turbo Decoding using the TVPE:
  - Turbo Coding for
    - 3GLTE (Evolved UTRA) turbo decoding as specified in 3GPP TS 36.212 (and 36.211), section 5.1.2.2
    - WiMAX OFDMA turbo decoding as specified in the **IEEE®** 802.16-2004™ standard, section 8.4.9.2
    - WiMAX OFDMA turbo decoding as specified in the IEEE® 802.16m-09/0010r2 standard, section 15.3.12
    - 3GPP TS 25.212 Multiplexing and channel coding
    - 3GPP2 (CDMA2000) standard, as specified in 3GPP2 C.S0002 rev A,B, C, D Physical Layer Standard for cdma2000 Spread Spectrum Systems, Section 2.1.3.1.4.2
    - TD-SCDMA standard (defined similarly to 3GPP TS 25.212 Multiplexing and channel coding)
  - Encoding Rates 1/3 and 1/5
  - K = 4 Turbo Codes
  - Soft and Hard Outputs
  - Zero Tailing and Tail biting trellis termination
  - Max Log Map with Non Linear Dynamic extrinsic Factorization
  - Hybrid Linear Log Map (MAX*) with programmable Linear Correction factor
  - Maximum decoded block size of:
    - 6144 for 3GLTE
    - 4800 for WiMAX
    - 5114 for 3GPP
    - 20,730 for 3GPP2

Note: For block sizes larger than 8 K (for rate 1/3) or larger than 4 K (for rate 1/5), the MAPLE-B needs special treatment for the input data structure. For details see **Section 26.3.1.4.7.1**, *Zero Tail Identification for Separate Vectors*, on page 26-37.

  - Programmable CRC check on Hard Output results
  - Aposteriori Quality (AQ) based and CRC based Stopping conditions
  - Programmable maximum/minimum iteration counters
  - Programmable number of internal Turbo engines
  - Programmable ascending or descending bit or/and byte ordering for output data

— Channel Data Stream Processor that provides “on-the-fly” Rate De-Matching (3GPP TS 25.212 Multiplexing and channel coding, section 4.8) and Sub-Block de-interleaving
— Support for multiple encoded input data structures
— Ideal 157 Mbps (running at 450 MHz) Turbo Decoding (8 iterations using direct data structure only)

Note: Because Turbo and Viterbi processing use the same PE, the ideal performance value assumes only Turbo decoding is performed.

- Viterbi Decoding using the TVPE:
  — Convolutional Coding support as specified in for:
    - 3GLTE (Evolved UTRA) decoding as specified in 3GPP TS 36.212, section 5.1.2.2
    - WiMAX OFDMA decoding as specified in the **IEEE** 802.16-2004 standard and corrigendum **IEEE** P802.16-2004/Cor2/D2, section 8.4.9.1
    - 3GPP TS 25.212 Multiplexing and channel coding
    - 3GPP2 (CDMA2000) standard, as specified in 3GPP2 C.S0002 rev A,B, C, D Physical Layer Standard for cdma2000 Spread Spectrum Systems, Section 2.1.3.1.4.1
    - TD-SCDMA standard (defined similarly to 3GPP TS 25.212 Multiplexing and channel coding
    - GSM (EDGE) standard
  — Encoding Rates of 1/2, 1/3 and 1/4
  — Decoding Convolutional codes with K= 9, 8, 7, 6, 5
  — Maximum internal block sizes, as follows:
    - 8192 (K=5)
    - 5376 (K=6)
    - 2688 (K=7)
    - 1344 (K=8)
    - 672 (K=9)

Note: The maximum size include the Zero Tail bit in case of Zero Tail decoding. For details, see **Section 26.3.2.1.2**, *Viterbi Decoding Operation*, on page 26-64.

— Support for decoding blocks up to 32K bits using a multiple Buffer Descriptors scheme (see **Section 26.3.2.1.2.4**, *Viterbi Large Blocks Partitioning Support*, on page 26-69).
— Zero tail decoding support
— Tail biting decoding support using WAVA* (Wrap-Around-Viterbi-Algorithm) (see **Section 26.3.2.1.2.3**, *Tail Biting Viterbi Processing (WAVA*)*)
— Multiple encoded input data structures support

- Periodic De-Puncturing scheme using a Channel Data Stream Processor for supporting various rate cases (for example, 1/3, 1/2, 2/3, 3/4, 5/6)
- Ascending and descending bit and/or byte ordering for output data
- Throughput (because Turbo and Viterbi processing are done using the same PE, the given performance assumes only Viterbi decoding is performed):
  - Ideal 100 Mbps Viterbi Decoding (K=7, tail biting, 2 iterations) running at 450 MHz
  - Ideal 112 Mbps Viterbi Decoding (K=9, zero tailed) running at 450 MHz

■ FFT/iFFT Processing using the FFTPE:
- Variable length FFT/iFFT processing of 128, 256, 512, 1024, and 2048 points
- FFT processing rates up to 280 Msps running at 450 MHz
- Programmable guard band insertion for iFFT
- Pre-Multiplication support by different complex value for each FFT sample
- Programmable scaling method, supporting one of the following methods:
  - Automatic adaptive scaling using overflow detection between transform stages, and possible programmable overall scaling factor for the output data
  - User defined scaling between transform stages
- Programmable input scale up for scaled down input data

■ DFT/iDFT Processing (DFTPE) with the following features:
- Variable length DFT/iDFT processing of the form $2^k \cdot 3^m \cdot 5^n \cdot 12$, up to 1536 points: 12, 24, 36, 48, 60, 72, 96, 108, 120, 144, 180, 192, 216, 240, 288, 300, 324, 360, 384, 432, 480, 540, 576, 600, 648, 720, 768, 864, 900, 960, 972, 1080, 1152, 1200 and 1536 points
- DFT processing at rates up to 180 Msps at 450 MHz
- Optional FFT/iFFT processing of 128, 256, 512, and 1024 points with throughputs up to 175 Msps at 450 MHz

Note: The guard band insertion feature is supported for FFT processing in the DFTPE as well.

- Programmable scaling method that supports one of the following methods:
  - Automatic adaptive scaling using overflow detection between transform stages, and possible programmable overall scaling factor for the output data
  - User defined scaling between transform stages
- Programmable input scale up for scaled down input data

■ CRC processing with the following features:
- Four possible polynomials
- CRC check or CRC calculation for block sizes of up to 128 Kb
- Optional byte reverse CRC processing
- CRC processing with throughput of up to 9.5 Gbps at 450 MHz

Note: If TVPE stop criteria is enabled, the CRCPE can work with throughputs of up to 10 Gbps only.

- Buffer Descriptor (BD) based programming model:
  - Up to 8 High Priority BD rings and 8 Low Priority BD rings for each processing element for multiple master support
  - 12 KB of BD rings internal memory
  - TaskID for every BD
- Flexible interrupt scheme
  - Up to 16 maskable BD ring interrupts
  - Programmable allocation of BD rings to interrupts
  - 2 (Program/Data) ECC non maskable interrupts from PSIF.
  - 2 Data ECC non-maskable interrupts from DFTPE/FFTPE
  - 1 System error maskable interrupt
  - Serial RapidIO Door-bell interrupt assertion support for External masters
- Power saving support during idle periods
- 2 MBus master interfaces for data transfers to/from system memory with total throughput up to 50 Gbps
- 1 MBus slave for accessing MAPLE-B for configuration, control, and internal memories
- 1 SBus slave for PSIF control

26.2 Modes of Operation

The MAPLE-B supports several modes of operation. These operation modes are Turbo related only. The Viterbi and DFT/FFT processing are not operation mode related and are fully configurable by the host. Activating the operation modes is done during the activation sequence of the MAPLE-B using its API.

26.2.1 3GLTE Standard Operation Mode

The MAPLE-B internal parameters and configuration registers are configured according to the 3GLTE standard. In this operation mode, the MAPLE-B can execute Turbo processing according to the 3GLTE standard only.

26.2.2 WiMAX Standard Operation Mode

The MAPLE-B internal parameters and configuration registers are configured according to the WiMAX standards. In this operation mode, the MAPLE-B can execute Turbo processing according to the WiMAX standards only.

26.2.3 3GPP Standard Operation Mode

The MAPLE-B internal parameters and configuration registers are configured according to the 3GPP standard. In this operation mode, the MAPLE-B can execute Turbo processing according to the 3GPP standard only.

26.2.4 3GPP2 Standard Operation Mode

The MAPLE-B internal parameters and configuration registers are configured according to the 3GPP2 standard. In this operation mode, the MAPLE-B can execute Turbo processing according to the 3GPP2 standard only.

26.3 Functional Description

The MAPLE-B functional description is divided into the following three areas.

- Buffer descriptors. General to all Accelerators (TVPE, FFTPE, DFTPE, and CRCPE).
- Accelerator specific information for TVPE, FFTPE/DFTPE, and CRCPE.
- MAPLE system interfaces. Used by all PEs.

26.3.1 Buffer Descriptors (BDs)

The logical handshake between the DSP device cores and peripherals and the MAPLE-B is buffer-descriptor based. Up to 8 hosts are supported for each PE, with each host receiving one high-priority and one low-priority buffer descriptors ring. Depending on the configuration, up to 16 hosts are supported for each PE, with each host receiving one buffer descriptors ring with the same priority assigned to all the rings. The MBDRCP parameter controls the arbitration order between the high and low rings. For details, see **Section 26.3.1.2**, *BD Arbitration*.

Note: There is no arbitration between the PEs, because each PE is targeted for a different type of processing, and the job allocation of the PEs is performed according to the availability of each PE.

26.3.1.1 BD Rings Arbitration

The MAPLE-B manages its buffer-descriptor rings using 4 sets of configuration parameters for each PE:

- **M<pe>BRHPA$_x$P**. Defines the first set of attributes for High Priority Buffer-Descriptor ring x.
- **M<pe>BRHPB$_x$P.** Defines the second set of attributes for High Priority Buffer-Descriptor ring x.
- **M<pe>BRLPA$_x$P**. Defines the first set of attributes for Low Priority Buffer-Descriptor ring x.

- **M<pe>BRLPB$_x$P.** Defines the second set of attributes for Low Priority Buffer-Descriptor ring x.

Note: <pe> stands for the specified PE: TV for TVPE; FF for FFTPE; DF for DFTPE, and CRC for CRCPE.
x stands for the number of parameters for each type (x = 0 to 7).

Each set of parameters includes the following attributes for each BD-ring (high or low):

- BDR_BA—Base address for each of the BD-rings in the dedicated buffer descriptor memory. The BD ring base address must be aligned to a 256 bytes address, and can only be changed if the valid bit ([VLD]) of the ring configuration parameter is disabled.
- BDR_RD_PTR—Pointer to a BD in the BD-ring, which is the current BD job being processed. This pointer is post increment by MAPLE-B once the BD job is complete. This field should be initialized by the host whenever the [BDR_BA] field is updated and with the same value as the [BDR_BA]. It must be done while the valid bit ([VLD]) of the ring descriptor is disabled.This field is used by MAPLE-B to locate the next BD in the ring. If the OWNER bit of the BD pointed by this field is cleared, the MAPLE-B assumes no jobs are available in the ring.
- INT_TRGT—Interrupt target. Describe which of the MAPLE-B BD rings interrupts (bdr_done0 to bdr_done15) is to be asserted on completion of a BD from this ring The.
- VLD—Valid bit. When set, the MAPLE-B checks the owner bit of the BD pointed to by the [BDR_RD_PTR] in this ring. When the [VLD] bit is set, no changes are allowed in the M<*pe*>BR(H/L)P(A/B)xP parameters.

The following three parameters are valid only when the master of the ring is external to the DSP device and the MAPLE-B must communicate with it using the serial RapidIO doorbell port. See **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell* for details.

- [HOST_ID]—Host ID in case of external master.
- [P_TR_IF]—Port Target Interface. Up to 16 possible RapidIO target interfaces. This field is used by the MAPLE-B for the *ODDATR[TGINT]* field in the serial RapidIO Doorbell controller.
- [EXT_MST]—Specifies whether the master is connected to MAPLE-B via regular interrupt or Serial RapidIO Doorbell interrupt.

26.3.1.2 BD Arbitration

Having multiple BD rings with multiple priority data structures creates the need for arbitration between the following elements:

- High BD rings queue and low BD rings queue.
- Rings in the high/low BD rings.
- BDs in each of the rings.

The first level of arbitration uses the MBDRCP[<*pe*>_HL_ARB] (<*pe*> = TV, FF, DF). This field describes the arbitration order between the high BD rings and the low BD rings for each PE. It can be configured to scan the high BD rings only, to scan high/low BD rings equally (thus allowing up to 16 masters with each receiving one BD ring), or to scan the high BD rings more often than the low BD rings. See **Section 26.4.2.1.1**, *MAPLE BD Rings Configuration Parameter (MBDRCP)*. For example, if the chosen arbitration is H-H-L, the MAPLE-B searches for the first BD in the first valid high priority ring, the next BD is taken from the second valid high priority ring (if it does not exist then the following BD in the same high priority ring). The third BD is taken from the first low priority ring and the forth BD is taken from the next high priority ring again. It continues henceforth.

The second level of arbitration is a simple cyclic arbitration between the rings of the same priority. For example, if during a certain configuration of a certain PE, low priority rings #0, #1 and #3 are valid, the MAPLE-B executes BDs according to the following order: when the low priority rings are accesses, the MAPLE-B executes one BD from ring#0. On the next visit to the low priority rings it executes one BD from ring#1 and on the next visit one BD from ring#3 and so on. If during a certain visit in the certain ring there is no BD to process, the MAPLE-B continues to the next ring according to its cyclic order.

The MAPLE-B uses the MBDRCP[pe_R_LOOP] (pe = TV,FF,DF,CRC) to determine the potential valid rings to be scanned in each ring queue. For example, if MBDRCP[TV_R_LOOP] equals 4, the MAPLE-B will search for valid rings only in ring #0 to ring #4 in its high/low ring queues.

Note: Allowing more potential rings than actually used causes the internal RISC engines to expand their search to more rings than required thus degrading the MAPLE-B performance.

Within each BD ring, MAPLE-B expects to find its next valid BD according to the M<pe>BR(H/L)PAxP[BDR_RD_PTR] field in the DRAM. If the OWNER bit of that BD is set, the MAPLE-B starts executing this BD (assuming the relevant PE is available). If the OWNER bit is not set, the MAPLE-B does not execute any BD from the ring and jumps to the next ring as described above.

Each BD (regardless of its type) includes a [WRAP] bit. This bit tells MAPLE-B how to update the M<pe>BR(H/L)PAxP[BDR_RD_PTR] upon job completion. If the [WRAP] bit is not set then M<pe>BR(H/L)PAxP[BDR_RD_PTR] is increment with the BD size, that is, the next BD address is the current BD address plus the BD size. If the [WRAP] bit is set, then MAPLE-B updates the M<pe>BDR(H/L)PAxP[BDR_RD_PTR] with the value of M<pe>BR(H/L)PAxP[BDR_BA], that is, it expects to find the next BD at the base address of the current BD ring.

Note: The host must ensure that the last BD in its ring has the WRAP bit set. A state of one active BDR with single circular BD ([WRAP] bit is set) is forbidden for any of the PEs. Such a state may result in an unknown state of the MAPLE-B.

BD handshake with external masters uses the [OWNER] bit. Each BD (regardless of its type) includes an [OWNER] bit. If the [OWNER] bit of the BD is set, MAPLE-B starts executing this BD. On completion, it clears the [OWNER] bit and increments the M<pe>BR(H/L)PAxP[BDR_RD_PTR] as described above. If the [OWNER] bit is not set MAPLE-B continues to the next BD ring according to its arbitration scheme.

If no jobs are found in any of the BD-rings, or all of the accelerators are busy, the MAPLE-B hibernates and does not seek for new jobs. It wakes up due to one of the following reasons:

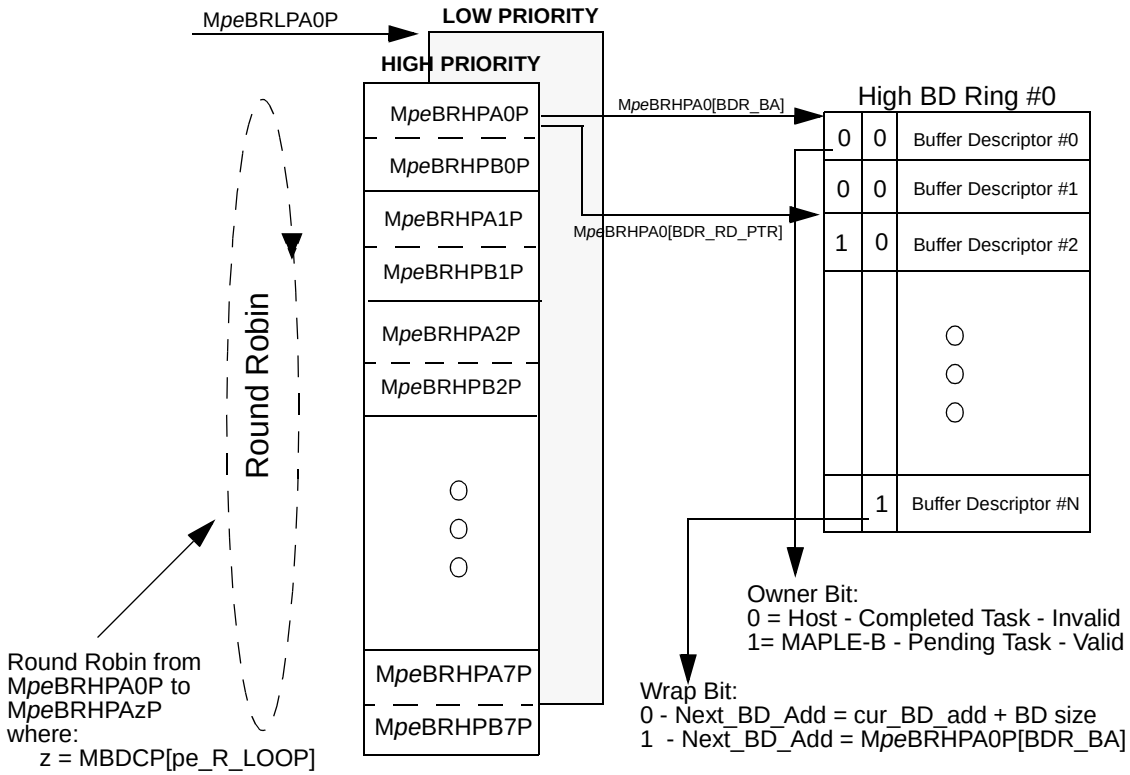
- Internal counter time-out. For power consideration, the MAPLE-B implements an internal dynamic time-out counter that is activated internally and wakes the internal engine periodically to scan for new BDs.
- A host accessing the PCR, as described in **Section 26.4.3.1**, *PSIF Command Register (PCR),* on page 26-169.
- Internal events from one of the PEs, such as job complete.

When a new BD job is found and the required accelerator is idle, the MAPLE-B proceeds with the BD execution. For details on job execution, refer to the specific accelerator used.

- **Section 26.3.1.4**, *Turbo/Viterbi Decoding Flow*, on page 26-11.
- **Section 26.3.2.2**, *FFT/iFFT/DFT/iDFT Operation*, on page 26-80.
- **Section 26.3.2.3**, *CRC Operation*, on page 26-107

When a job is completed, the MAPLE-B clears the [OWNER] field, and if specified in the accelerator-specific BD, initiates an interrupt to the host. The M<pe>BDR(H/L)PBxP[INT_TRGT] field is used to define the target of this interrupt, with up to 16 different target interrupts possible. If the M<pe>BRHPBxP[EXT_MST] is set to SoC external master, the MAPLE-B initiates a door-bell interrupt via the serial RapidIO port (see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell* for details).

Figure 26-2 illustrates the BD rings data structure and the mechanism of one PE in the PSIF DRAM of the MAPLE-B.



Note: The figure describes arbitration scheme of one PE only. There cannot be two different types of <*pe*> in one ring or queue of rings

Figure 26-2. MAPLE-B Buffer Descriptor Data Structure and Mechanism of One PE

26.3.1.3 BDs Data Coherency Bit

Each of the MAPLE-B BDs includes a Data Coherency (DC) bit indicating whether the MAPLE-B can override the OWNER bit and shift to the next BD before all the result data of the current BD is in the system memory or must wait until all result data of the current BD is in the system memory before shifting to the next BD. Allowing non Data Coherency means that the MAPLE-B overrides and clears the OWNER bit once the internal DMA receives all output data configurations and then continues to the next BD execution without waiting for internal DMA confirmation that all data has reached the system memory. If a host is polls the OWNER bit to ensure BD completion, then the DC bit must be set.

If a MAPLE-B interrupt is required at the end of the BD (INT_EN is set), the DC bit is ignored and the MAPLE-B clears the OWNER bit and asserts the interrupt only after all data transfer related to that BD is completed.

Note: Allowing the MAPLE-B to work without data coherency may slightly improve the throughput performance. It is best to allow several BDs to operate with no data coherency followed by a BD that requires data coherency by either setting the DC bit or the INT_EN bit. This ensures that when the OWNER bit of last BD is cleared, all BDs in that ring are completed and all the relevant data is in the system memory.

26.3.1.4 Turbo/Viterbi Decoding Flow

The MAPLE-B supports Turbo and Viterbi decoding for variable standards and configurations as described in **Section 26.1**, *Features*. Both the Turbo and Viterbi processing are done using the TVPE. Therefore, at given point in time the TVPE can either process Turbo decoding or Viterbi decoding. There is no parallel processing of Turbo and Viterbi. The MAPLE-B initialization sequence selects the desired Turbo standard operation mode (see **Section 26.2**, *Modes of Operation*).

26.3.1.4.1 Turbo Standard Parameter Assumptions

The MAPLE-B supports Turbo decoding according to the following standards (MAPLE-B operation modes): 3GLTE, WiMAX, 3GPP and 3GPP2. **Table 26-1** lists some of the parameters that the MAPLE-B uses for each of its operation modes.

Table 26-1. MAPLE-B Assumptions for Each Supported Standard

Operation Mode (Standard)	Turbo Rate	Turbo Trellis Termination Method	Turbo Encoding Method	Turbo Polynomials
3GLTE	1/3	Zero tail	Binary encoding	$g_0 = 1 + D^2 + D^3$ (feedback) $g_1 = 1 + D + D^3$ (parity)
WiMax	1/3	Tail biting	Duo-binary encoding	$g_0 = 1 + D + D^3$ (feedback) $g_1 = 1 + D^2 + D^3$ (Y parity) $g_2 = 1 + D^3$ (W parity)
3GPP	1/3	Zero tail	Binary encoding	$g_0 = 1 + D^2 + D^3$ (feedback) $g_1 = 1 + D + D^3$ (parity)
3GPP2	1/5	Zero tail	Binary encoding	$g_0 = 1 + D^2 + D^3$ (feedback) $g_1 = 1 + D + D^3$ (Y_0 parity) $g_2 = 1 + D + D^2 + D^3$ (Y_1 parity)

26.3.1.4.2 TVPE Buffer-Descriptor Structure

Figure 26-3 shows a TVPE Buffer Descriptors structure as it should be written to the TVPE BD rings by the host. Some of the fields may not be relevant for some configurations and are ignored by MAPLE-B. Nonrelevant fields can remain unwritten by the host.

Offset BD_BASE + 0x0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	OWNER	WRAP	DC	INT_EN	L_MAP	—	VIT_K			TBZE	ZTTB	CRC_EN	ENC_RATE		ADJ	ALG
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUNC_SCHM				MAX_ITER				MIN_ITER				IN_D_STRCT			

Offset BD_BASE + 0x4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W		BS														
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W				LLMAP_LCF					CRC_POLY		NDRE		W_16	PRF_TGL	Res.	

Offset BD_BASE + 0x8

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

Offset BD_BASE + 0xC

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

Figure 26-3. TVPE Buffer Descriptor Structure

Offset	BD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BASE_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BASE_ADDR															
W																

Offset	BD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADDR_OFFSET_0															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR_OFFSET_1															
W																

Offset	BD_BASE + 0x18															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADDR_OFFSET_2															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR_OFFSET_3															
W																

Offset	BD_BASE + 0x1C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADDR_OFFSET_4															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR_OFFSET_5															
W																

Offset	BD_BASE + 0x20															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFT_RSLT_ADDR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SFT_RSLT_ADDR															
W																

Figure 26-3. TVPE Buffer Descriptor Structure (continued)

Offset	BD_BASE + 0x24															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	E_PARAM_PTR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	E_PARAM_PTR															DREP
W																

Offset	BD_BASE + 0x28															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Offset	BD_BASE + 0x2C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Offset	BD_BASE + 0x30															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			CMP_RSN		ERR_ST				ZT_MAX				ITER_CNT			
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Offset	BD_BASE + 0x34															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Figure 26-3. TVPE Buffer Descriptor Structure (continued)

Offset	BD_BASE + 0x38															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Offset	BD_BASE + 0x3C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Figure 26-3. TVPE Buffer Descriptor Structure (continued)

Table 26-2. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Field	Description	Settings
OWNER 31	**Owner Bit** When set by the host, MAPLE-B is BD owner and the job is not complete. When cleared by MAPLE-B, the job is done and the host is the BD owner.	0 The host is BD owner 1 MAPLE-B is BD owner
WRAP 30	**Wrap Bit** Indicates to MAPLE-B where to find the next BD in the ring.	0 MAPLE-B expects to find the next BD at: MTVBR(H/L)PAxP[BDR_RD_PTR[13:6]] + BD size. 1 MAPLE-B updates MTVBR(H/L)PAxP[BDR_RD_PTR[13:6]] with MTVBR(H/L)PAxP[BDR_BA[13:8]], that is, it expects to find the next BD at the Base Address of the ring.
DC 29	**Data Coherency**. Valid only if INT_EN is reset. Indicates whether the MAPLE-B should maintain data coherency when clearing the OWNER bit. If set, the MAPLE-B assures Data transfers completion before clearing the OWNER bit, else it clears OWNER bit and shifts to next BD regardless of Data transfers status. For details see **Section 26.3.1.3**, *BDs Data Coherency Bit*.	0 No Data coherency is required. 1 Data coherency is required. The MAPLE-B assures Data transfer completion before clearing the OWNER bit and shifting to the next BD.
INT_EN 28	**Interrupt Enable** If set, MAPLE-B issues an interrupt/door-bell interrupt at the end of this job. If cleared no interrupt is issued.	0 End of job interrupt is NOT issued. 1 End of job interrupt is issued.
L_MAP 27	**LogMAP Algorithm** Valid only for Turbo decoding (ALG=0). Describes the LogMAP algorithm.	0 MaxLogMAP 1 Hybrid Linear LogMAP (Max*).

Table 26-2. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Field	Description	Settings
— 26	Reserved	
VIT_K[2:0] 25–23	**Viterbi Constraint Length (K)** Valid only for Viterbi decoding (ALG=1). Describes the encoding constraint length.Valid only during Viterbi decoding:	000 Reserved 001 K=5. 010 K=6. 011 K=7. 100 K=8. 101 K=9. 110 to 111 Reserved.
TBZE 22	**Trace Back from Zero Enable** Valid only for Viterbi decoding (ALG=1) and only if Zero Tail trellis termination is assumed (ZTTB=1). If set, the trace back calculation starts from Zero state, regardless of the maximum path calculations. If cleared the trace back calculation starts from the winning state of the maximum calculation regardless if the result was the Zero state.	0 The Trace back calculation always starts from the winner of the maximum path calculations. 1 The Trace Back calculation always starts from state zero.
ZTTB 21	**Zero Tail or Tail Biting Trellis Termination** **Note:** For Turbo decoding (ALG=1) it should be configured as Zero Tail for all operation modes excluding the WiMAX operation mode where it should be configured for Tail Biting.	0 Tail Biting trellis termination assumed. 1 Zero Tail trellis termination assumed.
CRC_EN 20	**CRC Check Enable** Valid only for Turbo decoding. If set the MAPLE-B is to run CRC check on the Hard Output results. The polynomial type is to be determined by MTCPCP[CRC_POLY]. The CRC results are described in the [ERROR_STATUS] field of the BD. If the MTSCCP[S_CRC] bit is set, that is, CRC stopping criteria is enabled, CRC check is executed regardless of this bit configuration. (see **Section 26.4.2.2.1**, *MAPLE- Turbo Stop Criteria Configuration Parameter(MTSCCP)*.	0 CRC check on the Hard Output is disabled. 1 CRC check on the Hard Output is enabled.
ENC_RATE[1:0] 19–18	**Turbo Decoding. Encoder Rate** Describes the encoding rate used for this job. Valid only for 3GPP2 operation mode. In all other Turbo standard operation modes it assumed as rate 1/3. **Note:** For Turbo decoding, the value of this field must apply with the assumptions described in **Table 26-1** Viterbi Decoding: Encoder Rate. Describes the Viterbi encoding rate used for this job.	00 Reserved. 01 1/3. 10 Reserved. 11 1/5. 00 1/2 01 1/3 10 1/4 11 Reserved.
ADJ 17	**Adjacent Bit** Valid only if Vectors input data structure (IN_D_STRCT=0,1, 7) is used. If set, MAPLE-B assumes all relevant input vectors are located adjacently in system memory.	0 MAPLE-B use the [BASE_ADDR] and [ADDR_OFFSET_x] fields of the BD to locate the vector buffers in system memory. 1 MAPLE-B assume all relevant vectors are adjacent in system memory. See **Section 26.3.1.4.12**, *Adjacent Bit Description ([ADJ])*

Table 26-2. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Field	Description	Settings
ALG 16	**Decoding Algorithm Type**	0 Turbo algorithm 1 Viterbi algorithm
PUNC_SCHM[3:0] 15–12	**Puncturing Scheme** Valid only for Periodically Punctured Configurable Mix Stream (PPCMS) input data structure (IN_D_STRCT=3). Selects one of the 10 possible de-puncturing schemes as described in MTVPVxHCP & MTVPVxLCP. See also **Section 26.3.1.4.8**, *Periodically Punctured Configurable Mix Stream (PPCMS) Data Structure*."	0000 Periodic De-puncture execution disabled. 0001 execute Periodic De-puncture scheme #0 0010 execute Periodic De-puncture scheme #1 ... 1010 execute Periodic De-puncture scheme #9 1011 to 1111 Reserved.
MAX_ITER[3:0] 11–8	**Maximum Number of Iterations** Valid only during Turbo decoding. Describes the Maximum number of full iterations to be executed assuming stop criteria was not reached. The MAX_ITER value is increment internally by one to represent the values 1 to 16. This value should always be larger than MIN_ITER.	0 to 15 maximum number of iterations 1 to 16, respectively.
MIN_ITER[3:0] 7–4	**Minimum Number of Iterations** Valid only during Turbo decoding. Describes the Minimum number of iterations executed by the MAPLE-B. If the stop criteria is reached before the minimum iteration value, the MAPLE-B continues decoding until this value is reached. This value should always be smaller or equal to MAX_ITER.	0 to 15 'minimum number of iterations 1 to 16, respectively
IN_D_STRCT[3:0] 3–0	**Input Data Structure** Describes the possible input data structures in system memory to be fetched by MAPLE-B into the TVPE.	0000 Direct. See **Section 26.3.1.4.6**, *Direct Data Structure*. 0001 Separate Vectors. See **Section 26.3.1.4.7**, *Separate Vectors Data Structure*. 0010 Reserved. 0011 Periodically Punctured Configurable Mix Stream (PPCMS). See.**Section 26.3.1.4.8**, *Periodically Punctured Configurable Mix Stream (PPCMS) Data Structure*. 0100 to 0101 Reserved. 0110 Rate-matched Fixed Mix Stream. See **Section 26.3.1.4.9**, *Rate-Matched Fixed Mix Stream Data Structure*. 0111 Sub-block Interleaved vectors. See **Section 26.3.1.4.10**, *Sub-Block Interleaved Vectors Data Structure*. 1000 to 1111 Reserved.

Table 26-3. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Field	Description	Settings
— 31	Reserved	
BS[14:0] 30–16	**Block Size** Describes the current tasks decoded block size in bits.	Valid values are 40 to 20,730 bits. This field should not include the tail bytes, if they exist.
— 15–13	Reserved	
LLMAP_LCF[4:0] 12–8	**Linear LogMAP Correction Factor** Valid only for Turbo decoding and only if Max* LogMAP algorithm is enabled (L_MAP=1). The value of this field must be even. For more details, see **Section 26.3.2.1.1.3**, *Turbo Processing Implementation*.	16 possible even decimal values.
CRC_POLY[1:0] 7–6	**CRC Polynomial** Describes which Polynomial to use if CRC check or CRC stop criteria is required.	00 CRC24 with the following Polynomial: $D^{24} + D^{23} + D^6 + D^5 + D + 1$ 01 CRC24 with the following Polynomial: $D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$ 10 CRC16-CCITT with the following polynomial: $D^{16} + D^{12} + D^5 + 1$ 11 CRC16 with the following polynomial: $D^{16} + D^{15} + D^{14} + D^{11} + D^6 + D^5 + D^2 + D + 1$
NDRE[1:0] 5–4	**Number of DRE Engines** Valid only during Turbo decoding. Determines the number of internal Turbo engines which participate in the decoding process. For more details see **Section 26.3.2.1.1.2**, *Number of Turbo Processing Elements (DREs)*.	00 Use the MAPLE-B's default number of DRE engines. 01 Use only one DRE. 10 Use two DRE engines 11 Use four DRE engines.
W_16 3	**WiMAX 16m Indication** Valid only during Turbo decoding (ALG=0) and only for the WiMAX operation mode. Indicates the MAPLE-B to configure the TVPE internal Interleavers parameter with the WiMAX 16m parameters.	0 The TVPE is configured with WiMAX IEEE® 802.16-2004™ standard 1 The TVPE is configured with WiMAX IEEE® 802.16m-09/0010r2 standard
PRF_TG 2	**Profiling Toggle Bit** If set the MAPLE-B changes its TVPE profiling state from enabled/disabled to disabled/enabled. See **Section 26.3.2.1.5.2**, *TVPE Profiling*.	
— 1–0	Reserved	

Table 26-4. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Field	Description	Settings
HOE 31	Hard Output Enable Controls the availability of hard-output results from the TVPE decoder. If set the MAPLE-B outputs the Hard Output results to the address in system memory as specified in HRD_RSLT_ADDR.	1 Hard output enabled 0 Hard output disabled
SOE[1:0] 30–29	Output Enable. Controls the availability, and type of the soft-output results from the decoder, applicable for Turbo decoding only.	00 No Soft Output 01 Aposteriori Output Enabled 10 Extrinsic Output Enabled. 11 Reserved
— 28	Reserved	
TASK_ID[7:0] 27–20	**Task ID** Task ID tag, supplied by host, assists in identification of its task when directly accessing the TVPE to inquire task status. When MAPLE-B assigns the job to the TVPE, it copies the TASK_ID into the TVPE, thus allowing the host to track the TVPE jobs.	
VIT_SET[1:0] 19–18	**Viterbi Set** Valid only for Viterbi decoding (ALG=1). Describes which set of Viterbi polynomial is used for the current job: For details, see **Section 26.4.4.1.6** and **Section 26.4.2.2.5**	00 MAPLE-B assume no configuration of the TVVPVG0CR & TVVPVG1CR registers is required. 01 MAPLE-B use MTVPVS1CxP parameters in order to configure the TVVPVG0CR & TVVPVG1CR registers. 10 MAPLE-B use MTVPVS2CxP parameters in order to configure the TVVPVG0CR & TVVPVG1CR registers. 11 MAPLE-B use MTVPVS3CxP parameters in order to configure the TVVPVG0CR & TVVPVG1CR registers.
BUF_SIZE[17:0] 17–0	**Buffer Size** Describes the actual input data buffer size in bytes if Periodically Punctured Configurable Mix Stream (PPCMS) or Rate-matched Fixed Mix Stream are enabled (IN_D_STRCT=3,6). The BUF_SIZE should include the tail bytes if enabled.	40B to 100KB are possible input block sizes.

Table 26-5. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Field	Description	Settings
HRD_RSLT_ADDR[31:3] 31–3	**Hard Result Address** Describes the base address in which the MAPLE-B provides Hard Output results, if enabled (HOE=1). The [HRD_RSLT_ARRD] field must be aligned to 8 bytes.	0x00000000 to 0xFFFFFFF8
— 2–0	Reserved	

Table 26-6. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x10

Field	Description	Settings
BASE_ADDR[31:6] 31–6	**Base Address** During the following configurations it points to the start address of the input buffer: • Periodically Punctured Configurable Mix Stream (PPCMS) or Rate-matched Fixed Mix Stream input data structures are enabled (IN_D_STRCT=3,6). • Any of the vectors input data structures (IN_D_STRCT=0,1,7) is enabled and the [ADJ] bit is set (1). During the following configurations it points to the base address in system memory to which all the ADDR_OFFSET_x fields are related: • Any of the vectors input data structures (IN_D_STRCT=0,1,7[1]) is enabled and the [ADJ] bit is reset (0). The [BASE_ADDR] field must be aligned to 64 bytes address.	0x0000000 to 0xFFFFFFF
— 5–0	Reserved	

1. For IN_D_STRCT=7 (Sub-Block Interleaved) in 3GLTE operation mode, there is only one input pointer (HARQ buffer), therefore it is always pointed by the BASE_ADDR.

Table 26-7. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Field	Description	Settings
ADDR_OFFSET_0[15:0] 31–16	**Buffer0 Address Offset**. Describes the upper 16 bits of a 22 bit relative offset address to BASE_ADDR if Vectors input data structure (IN_D_STRCT=0,1, 7) is enabled and ADJ=0. The Offset should point to a 64B aligned address. For details on the vector pointed to by this field according to the input data structure, see **Section 26.3.1.4.6**, *Direct Data Structure* to **Section 26.3.1.4.11**, *Supported Input Data Structures for Different Standards*.	0x0000 0 Bytes Offset 0x0001 64 Bytes Offset .... 0xFFFF 4096 KBytes Offset
ADDR_OFFSET_1[15:0] 15–0	**Buffer1 Address Offset** Describes the upper 16 bits of a 22 bit relative offset address to BASE_ADDR if Vectors input data structure (IN_D_STRCT=0, 1, 7) is enabled and ADJ=0. The Offset should point to a 64B aligned address. For details on the vector pointed to by this field according to the input data structure, see **Section 26.3.1.4.6**, *Direct Data Structure* to **Section 26.3.1.4.11**, *Supported Input Data Structures for Different Standards*.	0x0000 0 Bytes Offset 0x0001 64 Bytes Offset .... 0xFFFF 4096 KBytes Offset

Table 26-8. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Field	Description	Settings
ADDR_OFFSET_2[15:0] 31–16	**Buffer2 Address Offset** Describes the upper 16 bits of a 22 bit relative offset address to BASE_ADDR if Vectors input data structure (IN_D_STRCT=0, 1, 7) is enabled and ADJ=0. The Offset should point to a 64B aligned address. For details on the vector pointed to by this field according to the input data structure, see **Section 26.3.1.4.6**, *Direct Data Structure* to **Section 26.3.1.4.11**, *Supported Input Data Structures for Different Standards*.	0x0000 0 Bytes Offset 0x0001 64 Bytes Offset .... 0xFFFF 4096 KBytes Offset
ADDR_OFFSET_3[15:0] 15–0	**Buffer3 Address Offset** Describes the upper 16 bits of a 22 bit relative offset address to BASE_ADDR if Vectors input data structure (IN_D_STRCT=0, 1, 7) is enabled and ADJ=0. The Offset should point to a 64B aligned address. For details on the vector pointed to by this field according to the input data structure, see **Section 26.3.1.4.6**, *Direct Data Structure* to **Section 26.3.1.4.11**, *Supported Input Data Structures for Different Standards*.	0x0000 0 Bytes Offset 0x0001 64 Bytes Offset .... 0xFFFF 4,096 K-Bytes Offset

Table 26-9. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Field	Description	
ADDR_OFFSET_4[22:7] 31–16	**Buffer4 Address Offset** Describes the upper 16 bits of a 22 bit relative offset address to BASE_ADDR if Vectors input data structure (IN_D_STRCT=0, 1, 7) is enabled and ADJ=0. The Offset should point to a 64B aligned address. For details on the vector pointed to by this field according to the input data structure, see **Section 26.3.1.4.6**, *Direct Data Structure* to **Section 26.3.1.4.11**, *Supported Input Data Structures for Different Standards*.	0x0000 0 Bytes Offset 0x0001 64 Bytes Offset .... 0xFFFF 4,096 KBytes Offset
ADDR_OFFSET_5[22:7] 15–0	**Buffer5 Address Offset** Describes the upper 16 bits of a 22 bit relative offset address to BASE_ADDR if Vectors input data structure (IN_D_STRCT=0, 1, 7) is enabled and ADJ=0. The Offset should point to a 64B aligned address. For details on the vector pointed to by this field according to the input data structure, see **Section 26.3.1.4.6**, *Direct Data Structure* to **Section 26.3.1.4.11**, *Supported Input Data Structures for Different Standards*.	0x0000 0 Bytes Offset 0x0001 64 Bytes Offset .... 0xFFFF 4,096 K-Bytes Offset

Table 26-10. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x20

Field	Description	Settings
SFT_RSLT_ADDR[31:3] 31–3	**Soft Result Address** The address where the MAPLE-B provides the Soft Output results, if enabled (SOE= 1, 2). The [SFT_RSLT_ADDR] field must be aligned to 8 bytes.	0x00000000 to 0xFFFFFFF8
— 2–0	Reserved.	

Table 26-11. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x24

Field	Description	
E_PARAM_PTR[31:3] 31–3	**E Parameters Pointer** Valid only during Turbo decoding (ALG=0) in 3GPP standard operation mode and when input data structure is Rate-matched Fixed Mix Stream (IN_D_STRCT=6). A pointer to the E parameters data structure in the system memory used by the TVPE in order to perform the Rate De-Matching process. For more detail see **Section 26.3.1.4.9, *Rate-Matched Fixed Mix Stream Data Structure***	
— 2–1	Reserved	
DREP 0	**De-Repetition** Valid only during Turbo decoding (ALG=0) in 3GPP standard operation mode and when input data structure is Rate-matched Fixed Mix Stream (IN_D_STRCT=6). Indicates the MAPLE-B which Rate De-Matching process to execute	0 The TVPE performs De-puncturing process. 1 The TVPE performs De-repetition process.

Table 26-12. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x28

Field	Description
Reserved [31:0]	0xXX....[1]

1. 0xXX = any value

Table 26-13. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x2C

Field	Description
Reserved [31:0]	0xXX....[1]

1. 0xXX = any value

Table 26-14. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x30

Field	Description	Settings
— 31–30	Reserved	
CMP_RSN[1:0] 29–28	**Complete Reason** Valid after [OWNER] bit is cleared by MAPLE-B, and if the [ERROR_STATUS] field is not equal to 0011 or 0100 indicating the job was not executed.	00 Complete due to MAX_ITER value reached 01 Complete due to Aposteriori quality Stop Criteria 10 Complete due to CRC Stop Criteria 11 Reserved
ERR_ST[3:0] 27–24	**Task Error Status Code** Valid only after [OWNER] bit is cleared by MAPLE-B.	0000 No Error 0001 Reserved. 0010 CRC check on Hard Output results Failed 0011 BD Config Error. Wrong BS value. Job was not executed! 0100 BD Config Error. Wrong IN_D_STRCT. Job was not executed! 0101–1111: Reserved
ZT_MAX 23	**Zero Tail Max** Valid only during Viterbi Zero Tail decoding ([ALG]=1, [ZTTB]=1) and after [OWNER] bit is cleared by MAPLE-B. This bit is set by MAPLE-B only if the maximum calculation result, before the trace-back execution, did not point to state zero, as expected in the Viterbi Zero Tail algorithm.	
— 22–20	Reserved	
ITER_CNT[3:0] 19–16	**Iteration Count** Valid only during Turbo decoding and after [OWNER] bit is cleared by MAPLE-B. Status field which describes the actual number of full iterations executed for this job. The actual iteration number is ITER_CNT +1.	
— 15–0	Reserved	

Table 26-15. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x34

Field	Description
Reserved [31:0]	Reserved

Table 26-16. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x38

Field	Description
Reserved [31:0]	Reserved

Table 26-17. TVPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x3C

Field	Description
Reserved [31:0]	Reserved

26.3.1.4.3 Buffer Descriptor Special Notes

- When the host is writing a new BD to its ring it must be done “bottom-up”, that is, the [OWNER] bit must be the last write access to the PSIF DRAM. If several BDs are written to the same ring, only the OWNER bit of the first written BD must be the last to be written.
- Once the [OWNER] bit is set by the host, do not change the BD content. It should only be rewritten only after the [OWNER] bit is cleared by MAPLE-B.
- The status fields in the BD are valid only after the [OWNER] bit is cleared by the MAPLE-B.

26.3.1.4.4 TVPE Input Data Structures

This section describes all possible input data structures for Turbo/Viterbi processing supported by the MAPLE-B. **Table 26-18** explains the meaning of the acronyms used in this section.

Table 26-18. Symbol Identification Acronyms

Acronym	Explanation
SD	Systematic Data
SDa	First Systematic Data (WiMAX Dou-Binary)
SDb	Second Systematic Data (WiMAX Duo-Binary)
SD~	Interleaved Systematic Data
SDa~	First Interleaved Systematic Data (WiMAX Dou-Binary)
SDb~	Second Interleaved Systematic Data (WiMAX Dou-Binary)
PFa	First Parity byte for non-interleaved half iteration
PFb	Second Parity byte for non-interleaved half iteration
PFy	First Parity byte for non-interleaved half iteration (WiMAX Dou-Binary)
PFw	Second Parity byte for non-interleaved half iteration (WiMAX Dou-Binary)
PSa	First Parity byte for interleaved half iteration
PSb	Second Parity byte for interleaved half iteration
PSy	First Parity byte for interleaved half iteration (WiMAX Dou-Binary)
PSw	Second Parity byte for interleaved half iteration (WiMAX Dou-Binary)

Table 26-18. Symbol Identification Acronyms (Continued)

Acronym	Explanation
P0	Viterbi Rate 1/2 or 1/3 or 1/4 - first parity
P1	Viterbi Rate 1/2 or 1/3 or 1/4- second parity
P2	Viterbi Rate 1/3 or 1/4 - third parity
P3	Viterbi Rate 1/4 - fourth parity
BS	Block Size

Table 26-19 describes the expected type of data for each of the possible supported standards (operation modes), with respect to the algorithm (Turbo/Viterbi) and the encoding rate.

Table 26-19. Turbo/Viterbi Input Data Type

No.	MAPLE-B Operation Mode	Descriptor Fields		Input Data	
		ALG[1]	ENC_RATE[2]	Systematic	Parity
1	3GLTE,3GPP	0 = Turbo	1 = 1/3	SD	PFa,PSa
2	3GPP2	0 = Turbo	3 = 1/5	SD	PFa,PFb,PSa,PSb
3	WiMAX (Duo-Binary)	0 = Turbo	1 = 1/3	SDa,SDb	PFy,PFw,PSy,PSw
4	N/A[3]	1 = Viterbi	0 = 1/2	N/A[4]	P0,P1
5	N/A	1 = Viterbi	1 = 1/3	N/A	P0,P1,P2
6	N/A	1 = Viterbi	2 = 1/4	N/A	P0,P1,P2,P3

1. ALG is the Algorithm field in the TVPE buffer descriptor.See **Section 26.3.1.4.2**, *TVPE Buffer-Descriptor Structure*.
2. ENC_RATE is the Encoding Rate field in the TVPE buffer descriptor. See **Section 26.3.1.4.2**, *TVPE Buffer-Descriptor Structure*.
3. Viterbi configuration is not dependent on the standard operation mode.
4. For Viterbi, there is no systematic data, only parity bits.

The input data for each configuration as described in **Table 26-19**, is composed of input samples stored as 8-bit twos-complement numbers, and is arranged in system memory in the data structures described in **Section 26.3.1.4.6**, *Direct Data Structure*, **Section 26.3.1.4.7**, *Separate Vectors Data Structure*, **Section 26.3.1.4.8**, *Periodically Punctured Configurable Mix Stream (PPCMS) Data Structure*, **Section 26.3.1.4.9**, *Rate-Matched Fixed Mix Stream Data Structure*, and **Section 26.3.1.4.10**, *Sub-Block Interleaved Vectors Data Structure*.

26.3.1.4.5 Input Sample Polarity

As mentioned above, the input data samples to the TVPE are 8-bit 2’s complement soft numbers ranging from -128 to +127, configuration of which values (positive values or negative values) are to represent the logic 0 and the logic 1 of the input data is done during the mode parameter of the API (see **Section 26.3.4**, *Initialization Information*, on page 26-123). The two possible representations are:

logic 0 is mapped to -1 (-128 in 2’s complement representation)
logic 1 is mapped to +1 (+127 in 2’s complement representation)
or
logic 0 is mapped to +1 (+127 in 2’s complement representation)
logic 1 is mapped to -1 (-128 in 2’s complement representation)

26.3.1.4.6 Direct Data Structure

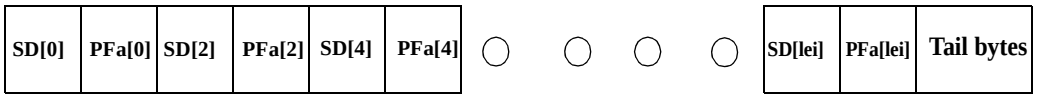
In Direct Data Structure, the MAPLE-B TVPE does not perform any kind of pre-processing on the input data. The data in system memory should be arranged in a unique data structure which fits the TVPE internal memory data structure. This data structure depends on the chosen standard operating mode (3GLTE/WiMAX/3GPP/3GPP2), algorithm (Turbo/Viterbi) and encoding rate (1/3,1/5 for Turbo and 1/2,1/3,1/4 for Viterbi). The following describes the expected data structures for all possible configurations as described in **Table 26-19**.

For Direct Data Structure, the IN_D_STRCT field of the TVPE BD must be equal to 0000.

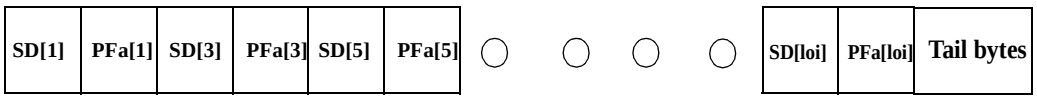
26.3.1.4.6.1 3GLTE and 3GPP Operation Modes—Rate 1/3 Turbo Processing

For these operation modes, the MAPLE-B expects to find the following vectors in system memory:

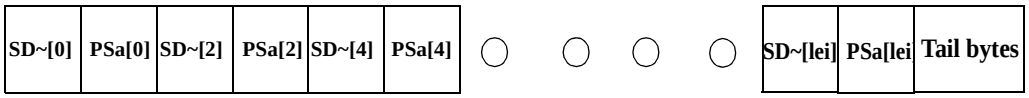
1. Even [index] number, interlaced SD and PFa bytes as follows:



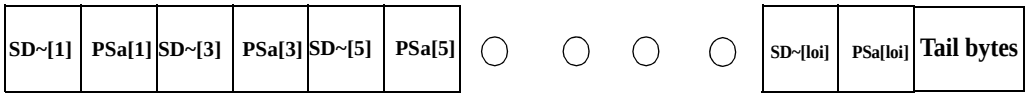
2. Odd [index] number, interlaced SD and PFa bytes as follows:



3. Even [index] number, interlaced SD~ and PSa bytes as follows:



4. Odd [index] number, interlaced SD~ and PSa bytes as follows:



where:

lei and loi stand for Last-Even-Index and Last-Odd-Index, respectively, and equal to:
If (BS is even)
lei = BS - 2

loi = BS - 1

Else

lei = BS - 1

loi = BS - 2

26.3.1.4.6.1.1 Tail Bytes

All the standards described in this sub-section use Zero Tail trellis termination, therefore Tail bytes should be included in the input data vectors. The Tail bytes should appear at the end of each vector to fit the TVPE internal data structure. The Tail bytes type, which are expected for this operation mode, are:

$SD_t[0]$, $SD_t[1]$, $SD_t[2]$, $SD\sim_t[0]$, $SD\sim_t[1]$, $SD\sim_t[2]$, $PFa_t[0]$, $PFa_t[1]$, $PFa_t[2]$, $PSa_t[0]$, $PSa_t[1]$, $PSa_t[2]$
where the subscript t indicates a Tail byte.

The order of these bytes in each vector depends on the current Block Size (BS field in the TVPE BD) parity- either an even number or an odd number.

If BS is an even number, the Tail bytes should be added to the vectors as follows:

Vector #1:	$SD_t[0]$	$PFa_t[0]$	$SD_t[2]$	$PFa_t[2]$
Vector #2:	$SD_t[1]$	$PFa_t[1]$	0x00	0x00
Vector #3:	$SD\sim_t[0]$	$PSa_t[0]$	$SD\sim_t[2]$	$PSa_t[2]$
Vector #4:	$SD\sim_t[1]$	$PSa_t[1]$	0x00	0x00

If BS is an odd number, the Tail bytes should be added to the vectors as follows:

Vector #1:			$SD_t[1]$	$PFa_t[1]$
Vector #2:	$SD_t[0]$	$PFa_t[0]$	$SD_t[2]$	$PFa_t[2]$
Vector #3:			$SD\sim_t[1]$	$PSa_t[1]$
Vector #4:	$SD\sim_t[0]$	$PSa_t[0]$	$SD\sim_t[2]$	$PSa_t[2]$

Note: If BS is even, the size of all four vectors before adding the Tail bytes is equal (each vector includes BS/2 [index] numbers). Therefore after adding the Tail bytes, 2 additional zero padding bytes are needed for all the vectors to be the same length. If BS is odd, no extra zero padding is needed since the even vectors (#1 and #3), before adding the Tail, include one more index than the odd vectors.

26.3.1.4.6.1.2 Examples for Vectors Generation:

1. If BS=327 then the vectors before adding Zero Tail should be:

Vector #1:	SD[0]	PFa[0]	○ ○ ○	SD[324]	PFa[324]	SD[326]	PFa[326]	
Vector #2:	SD[1]	PFa[1]	○ ○ ○	SD[325]	PFa[325]			
Vector #3:	SD~[0]	PSa[0]	○ ○ ○	SD~[324]	PSa[324]	SD~[326]	PSa[326]	
Vector #4:	SD~[1]	PSa[1]	○ ○ ○	SD~[325]	PSa[325]			

Since BS is odd number, the vectors after adding Tail bytes should look like:

Vector #1:	SD[0]	PFa[0]	○ ○ ○	SD[324]	PFa[324]	SD[326]	PFa[326]	$SD_t[1]$	$PFa_t[1]$
Vector #2:	SD[1]	PFa[1]	○ ○ ○	SD[325]	PFa[325]	$SD_t[0]$	$PFa_t[0]$	$SD_t[2]$	$PFa_t[2]$
Vector #3:	SD~[0]	PSa[0]	○ ○ ○	SD~[324]	PSa[324]	SD~[326]	PSa[326]	$SD\sim_t[1]$	$PSa_t[1]$
Vector #4:	SD~[1]	PSa[1]	○ ○ ○	SD~[325]	PSa[325]	$SD\sim_t[0]$	$PSa_t[0]$	$SD\sim_t[2]$	$PSa_t[2]$

2. If BS=1538 then the vectors before adding Zero Tail should be:

Vector #1:	SD[0]	PFa[0]	○ ○ ○	SD[1536]	PFa[1536]
Vector #2:	SD[1]	PFa[1]	○ ○ ○	SD[1537]	PFa[1537]
Vector #3:	SD~[0]	PSa[0]	○ ○ ○	SD~[1536]	PSa[1536]
Vector #4:	SD~[1]	PSa[1]	○ ○ ○	SD~[1537]	PSa[1537]

Since BS is even number, the vectors after adding Tail bytes should look like:

Vector #1:	SD[0]	PFa[0]	○ ○ ○	SD[1536]	PFa[1536]	$SD_t[0]$	$PFa_t[0]$	$SD_t[2]$	$PFa_t[2]$
Vector #2:	SD[1]	PFa[1]	○ ○ ○	SD[1537]	PFa[1537]	$SD_t[1]$	$PFa_t[1]$	*0x00*	*0x00*
Vector #3:	SD~[0]	PSa[0]	○ ○ ○	SD~[1536]	PSa[1536]	$SD\sim_t[0]$	$PSa_t[0]$	$SD\sim_t[2]$	$PSa_t[2]$
Vector #4:	SD~[1]	PSa[1]	○ ○ ○	SD~[1537]	PSa[1537]	$SD\sim_t[0]$	$PSa_t[0]$	*0x00*	*0x00*

26.3.1.4.6.1.3 BD Pointers Allocation

If the [ADJ] bit in the BD is cleared, initialize the correct vectors described above to the BD offset pointers (ADDR_OFFSET_x) as indicated in the table below:

Table 26-20. BD Offset Pointers Assignment for 3GLTE and 3GPP Rate 1/3 Direct Mode

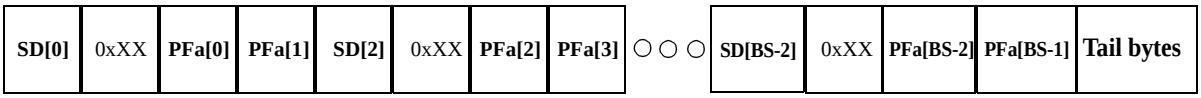
Offset Pointer	Vector Type
ADDR_OFFSET_0	Even index, interlaced SD & PFa (#1)
ADDR_OFFSET_1	Odd index, interlaced SD & PFa (#2)
ADDR_OFFSET_2	Even index, interlaced SD~ & PSa (#3)
ADDR_OFFSET_3	Odd index, interlaced SD~ & PSa (#4)
ADDR_OFFSET_4	N/A
ADDR_OFFSET_5	N/A

If the [ADJ] bit is set, the MAPLE-B expects to find the first vector in system memory in the address described in the [BASE_ADDR] field of the BD. The second vector should begin in the first aligned-to-8-bytes address after the end of the first vector and so on until the last vector. The order of the vectors should be: vector #1, vector #2,..., vector #4. For details see **Section 26.3.1.4.12**, *Adjacent Bit Description ([ADJ])*.

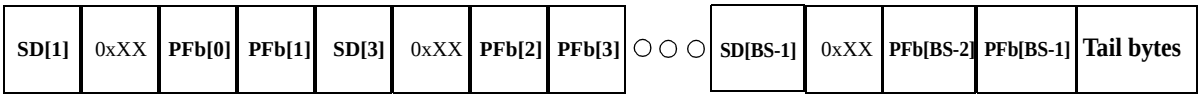
26.3.1.4.6.2 3GPP2 Operation Mode—Rate 1/5 Turbo Processing

For 3GPP2 operation mode with encoding rate of 1/5, the MAPLE-B expects to find the following vectors in system memory:

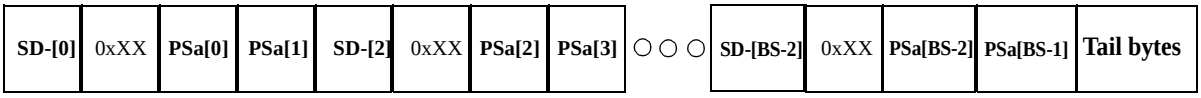
1. Interlaced SD and PFa bytes as follows:



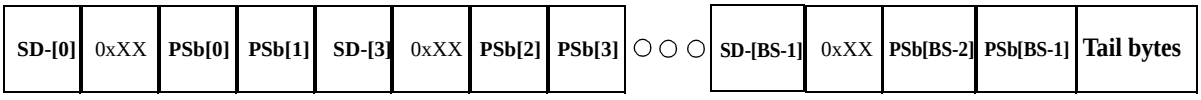
2. Interlaced SD and PFb bytes as follows:



3. Interlaced SD~ and PSa bytes as follows:



4. Interlaced SD~ and PSb bytes as follows:



where:

0xXX - one byte of any value

26.3.1.4.6.2.1 Tail Bytes

The 3GPP2-Rate-1/5 uses Zero Tail trellis termination, therefore, Tail bytes should be included in the input data vectors. The Tail bytes should appear at the end of each vector with additional zero padding to fit the TVPE internal data structure. The Tail bytes type that are expected for this operation mode are:

$SD_t[0]$, $SD_t[1]$, $SD_t[2]$, $SD\sim_t[0]$, $SD\sim_t[1]$, $SD\sim_t[2]$, $PFa_t[0]$, $PFa_t[1]$, $PFa_t[2]$, $PFb_t[0]$, $PFb_t[1]$, $PFb_t[2]$, $PSa_t[0]$, $PSa_t[1]$, $PSa_t[2]$, $PSb_t[0]$, $PSb_t[1]$, $PSb_t[2]$

where the subscript t indicates a Tail byte.

The expected order of these bits in each vector are as follows:

Vector #1:	$SD_t[0]$	0xXX	$PFa_t[0]$	$PFa_t[1]$	$SD_t[2]$	0xXX	$PFa_t[2]$	0x00
Vector #2:	$SD_t[1]$	0xXX	$PFb_t[0]$	$PFb_t[1]$	0x00	0xXX	$PFb_t[2]$	0x00
Vector #3:	$SD\sim_t[0]$	0xXX	$PSa_t[0]$	$PSa_t[1]$	$SD\sim_t[2]$	0xXX	$PSa_t[2]$	0x00
Vector #4:	$SD\sim_t[1]$	0xXX	$PSb_t[0]$	$PSb_t[1]$	0x00	0xXX	$PSb_t[2]$	0x00

26.3.1.4.6.2.2 Examples for Vectors Generation:

In the following examples, assume:

quad_i[x] represents:

if x is even	SD[x]	0xXX	PFa[x]	PFa[x+1]
if x is odd	SD[x]	0xXX	PFb[x-1]	PFb[x]

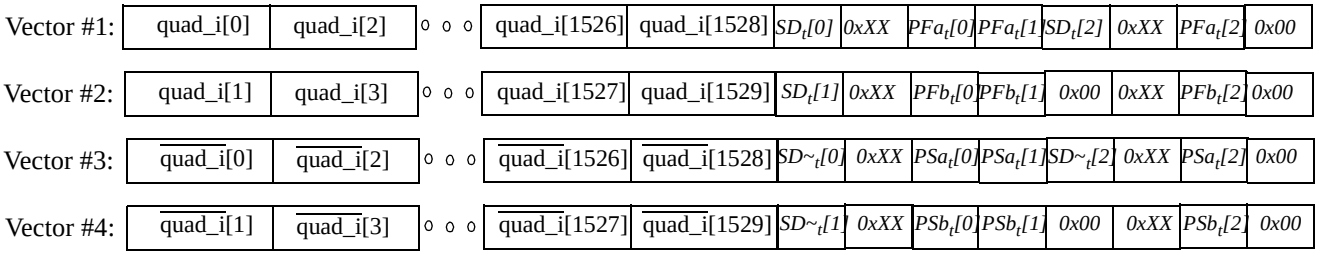
$\overline{\text{quad\_i}}$[x] represents:

if x is even	SD~[x]	0xXX	PSa[x]	PSa[x+1]
if x is odd	SD~[x]	0xXX	PSb[x-1]	PSb[x]

If BS=1530 then the vectors before adding Zero Tail should be:

Vector #1:	quad_i[0]	quad_i[2]	o o o	quad_i[1526]	quad_i[1528]
Vector #2:	quad_i[1]	quad_i[3]	o o o	quad_i[1527]	quad_i[1529]
Vector #3:	$\overline{\text{quad\_i}}$[0]	$\overline{\text{quad\_i}}$[2]	o o o	$\overline{\text{quad\_i}}$[1526]	$\overline{\text{quad\_i}}$[1528]
Vector #4:	$\overline{\text{quad\_i}}$[1]	$\overline{\text{quad\_i}}$[3]	o o o	$\overline{\text{quad\_i}}$[1527]	$\overline{\text{quad\_i}}$[1529]

The vectors after adding Tail bytes should look like:



26.3.1.4.6.2.3 BD Pointers Allocation

If the [ADJ] bit in the BD is cleared, initialize the correct vectors described above to the BD offset pointers (ADDR_OFFSET_x) as indicated in **Table 26-21**:

Table 26-21. BD Offset Pointers Assignment for 3GPP 2 Rate 1/5 Direct mode

Offset Pointer	Vector type
ADDR_OFFSET_0	Interlaced SD & PFa(#1)
ADDR_OFFSET_1	Interlaced SD & PFb(#2)
ADDR_OFFSET_2	Interlaced SD~ & PSa(#3)
ADDR_OFFSET_3	Interlaced SD~ & PSb (#4)
ADDR_OFFSET_4	N/A
ADDR_OFFSET_5	N/A

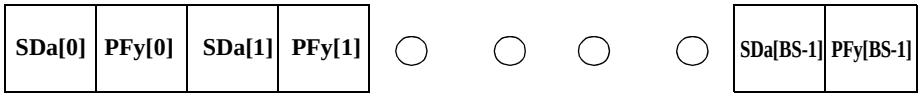
If the [ADJ] bit is set, the MAPLE-B expects to find the first vector in system memory, in the address as described in the [BASE_ADDR] field of the BD. The second vector should begin in the first aligned-to-8-bytes address after the end of the first vector and so on until the last vector. The order of the vectors should be: vector #1, vector #2,..., vector #4. For details see **Section 26.3.1.4.12**, *Adjacent Bit Description ([ADJ])*.

26.3.1.4.6.3 WiMAX Turbo Processing Operation Mode

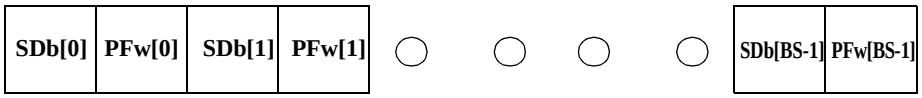
For this standard, the input data structure is different than used for the 3GLTE, 3GPP, and 3GPP2 because duo binary encoding is assumed. The encoding method is Tail biting and not Zero Tail, so no Tail bytes are expected at the end of the vectors.

A description of the four expected input data vectors for WiMAX operation mode and direct input data structure is as follows:

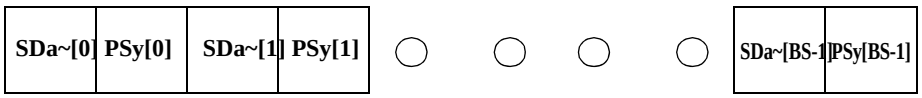
1. Interlaced SDa and PFy bytes:



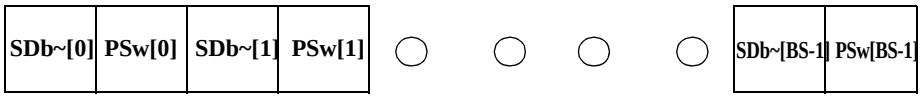
2. Interlaced SDb and PFw bytes:



3. Interlaced SDa~ and PSy bytes:



4. Interlaced SDb~ and PSw bytes:



26.3.1.4.6.3.1 BD Pointers Allocation

If the [ADJ] bit in the BD is cleared, initialize the correct vectors described above to the BD offset pointers (ADDR_OFFSET_x) as indicated in the table below:

Table 26-22. BD Offset Pointers Assignment for WiMAX Direct Mode

Offset Pointer	Vector type
ADDR_OFFSET_0	Interlaced SDa & PFy(#1)
ADDR_OFFSET_1	Interlaced SDb & PFw (#2)
ADDR_OFFSET_2	Interlaced SDa~ & PSy (#3)
ADDR_OFFSET_3	Interlaced SDb~ & PSw (#4)
ADDR_OFFSET_4	N/A
ADDR_OFFSET_5	N/A

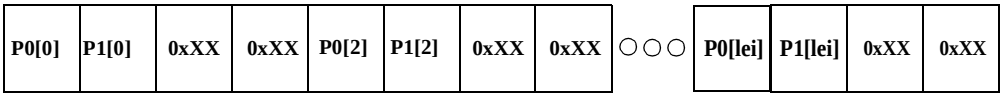
If the [ADJ] bit is set, the MAPLE-B expects to find the first vector in system memory, in the address as described in the [BASE_ADDR] field of the BD. The second vector should begin in the first aligned-to-8-bytes address after the end of the first vector and so on until the last vector. The order of the vectors should be: vector #1, vector #2,..., vector #4.

26.3.1.4.6.4 Viterbi Processing

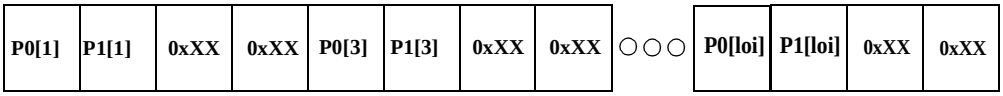
For Viterbi processing, the MAPLE-B expects to find two vectors for each encoding rate. The data structure of those vectors depends on the encoding rate. The data structures for each rate type are described in the following subsections.

26.3.1.4.6.4.1 Viterbi Rate 1/2:

1. Even [index] number, interlaced P0 and P1:

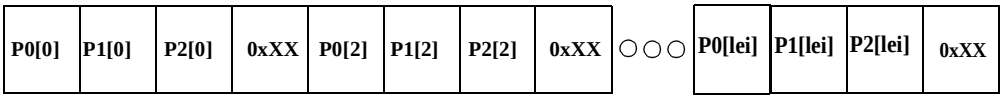


2. Odd [index] number, interlaced P0 and P1:

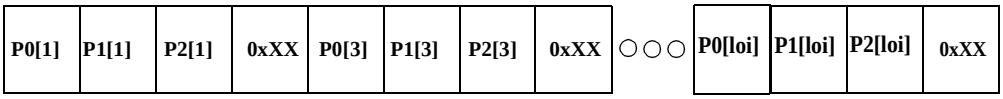


26.3.1.4.6.4.2 Viterbi Rate 1/3:

1. Even [index] number, interlaced P0, P1 and P2:

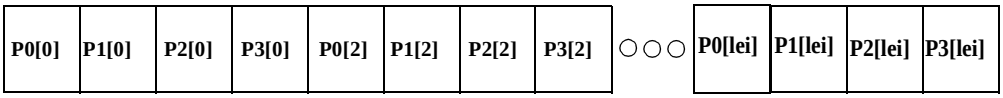


2. Odd [index] number, interlaced P0, P1, and P2:

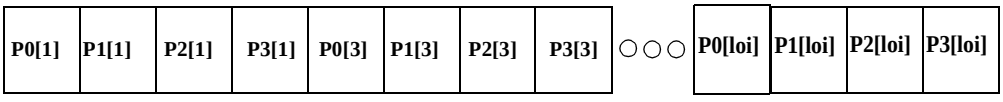


26.3.1.4.6.4.3 Viterbi Rate 1/4:

1. Even [index] number, interlaced P0, P1, P2, and P3:



2. Odd [index] number, interlaced P0, P1, P2, and P3:



where:

lei and loi stand for Last-Even-Index and Last-Odd-Index, respectively, and equal to:

If (BS is even)

lei = BS - 2

loi = BS - 1

Else

lei = BS - 1

loi = BS - 2

26.3.1.4.6.4.4 Zero Tail:

If Viterbi Zero Tail decoding configuration is set, The MAPLE-B expects to find the Tail bytes at the end of the vectors. The number of Tail bytes depends on the encoding rate and the Viterbi K parameter as described in the following equation: *Num of Tail bytes = (K-1) x (1/Rate)*. For example, for K=9, rate 1/3, there are 24 Tail bytes.

The structure of the Tail bytes at the end of the vectors depends if the decoded Block Size ([BS] field in the TVPE BD) is even or odd.

If the BS value is even then the structure of the Tail bytes at the end of each vector are as follows:

For rate 1/2 Even [index] vector, the Tail bytes at the end of the vector are:

$P0_t[0]$	$P1_t[0]$	0xXX	0xXX	$P0_t[2]$	$P1_t[2]$	0xXX	0xXX	○ ○ ○	$P0_t[N-2]$	$P1_t[N-2]$	0xXX	0xXX

For rate 1/2 Odd [index] vector, the Tail bytes at the end of the vector are:

$P0_t[1]$	$P1_t[1]$	0xXX	0xXX	$P0_t[3]$	$P1_t[3]$	0xXX	0xXX	○ ○ ○	$P0_t[N-1]$	$P1_t[N-1]$	0xXX	0xXX

For rate 1/3 Even [index] vector, the Tail bytes at the end of the vector are:

$P0_t[0]$	$P1_t[0]$	$P2_t[0]$	0xXX	$P0_t[2]$	$P1_t[2]$	$P2_t[2]$	0xXX	○ ○ ○	$P0_t[N-2]$	$P1_t[N-2]$	$P2_t[N-2]$	0xXX

$P0_t[0]$, $P1_t[0]$, $P2_t[0]$, 0xXX, $P0_t[2]$, $P1_t[2]$, $P2_t[2]$, 0xXX,..., $P0_t[N-2]$, $P1_t[N-2]$, $P2_t[N-2]$,0xXX

For rate 1/3 Odd [index] vector, the Tail bytes at the end of the vector are:

$P0_t[1]$	$P1_t[1]$	$P2_t[1]$	0xXX	$P0_t[3]$	$P1_t[3]$	$P2_t[3]$	0xXX	○ ○ ○	$P0_t[N-1]$	$P1_t[N-1]$	$P2_t[N-1]$	0xXX

For rate 1/4 Even [index] vector, the Tail bytes at the end of the vector are:

$P0_t[0]$	$P1_t[0]$	$P2_t[0]$	$P3_t[0]$	$P0_t[2]$	$P1_t[2]$	$P2_t[2]$	$P3_t[2]$	○ ○ ○	$P0_t[N-2]$	$P1_t[N-2]$	$P2_t[N-2]$	$P3_t[N-2]$

For rate 1/4 Odd [index] vector, the Tail bytes at the end of the vector are:

$P0_t[1]$	$P1_t[1]$	$P2_t[1]$	$P3_t[1]$	$P0_t[3]$	$P1_t[3]$	$P2_t[3]$	$P3_t[3]$	○ ○ ○	$P0_t[N-1]$	$P1_t[N-1]$	$P2_t[N-1]$	$P3_t[N-1]$

If the BS value is odd then before adding the Tail bytes, the size of the Even [index] vector is bigger by one [index] (4 bytes) than the size of the Odd [index] vector. The structure of the Tail bytes at the end of each vector is as follows:

For rate 1/2 Even [index] vector, the Tail bytes at the end of the vector are:

$P0_t[1]$	$P1_t[1]$	0xXX	0xXX	$P0_t[3]$	$P1_t[3]$	0xXX	0xXX	○ ○ ○	$P0_t[N-1]$	$P1_t[N-1]$	0xXX	0xXX

For rate 1/2 Odd [index] vector, the Tail bytes at the end of the vector are:

$P0_t[0]$	$P1_t[0]$	0xXX	0xXX	$P0_t[2]$	$P1_t[2]$	0xXX	0xXX	○ ○ ○	$P0_t[N-2]$	$P1_t[N-2]$	0xXX	0xXX

For rate 1/3 Even [index] vector, the Tail bytes at the end of the vector are:

$P0_t[1]$	$P1_t[1]$	$P2_t[1]$	0xXX	$P0_t[3]$	$P1_t[3]$	$P2_t[3]$	0xXX	○ ○ ○	$P0_t[N-1]$	$P1_t[N-1]$	$P2_t[N-1]$	0xXX

For rate 1/3 Odd [index] vector, the Tail bytes at the end of the vector are:

$P0_t[0]$	$P1_t[0]$	$P2_t[0]$	0xXX	$P0_t[2]$	$P1_t[2]$	$P2_t[2]$	0xXX	○ ○ ○	$P0_t[N-2]$	$P1_t[N-2]$	$P2_t[N-2]$	0xXX

For rate 1/4 Even [index] vector, the Tail bytes at the end of the vector are:

$P0_t[1]$	$P1_t[1]$	$P2_t[1]$	$P3_t[1]$	$P0_t[3]$	$P1_t[3]$	$P2_t[3]$	$P3_t[3]$	○ ○ ○	$P0_t[N-1]$	$P1_t[N-1]$	$P2_t[N-1]$	$P3_t[N-1]$

For rate 1/4 Odd [index] vector, the Tail bytes at the end of the vector are:

$P0_t[0]$	$P1_t[0]$	$P2_t[0]$	$P3_t[0]$	$P0_t[2]$	$P1_t[2]$	$P2_t[2]$	$P3_t[2]$	○ ○ ○	$P0_t[N-2]$	$P1_t[N-2]$	$P2_t[N-2]$	$P3_t[N-2]$

Where:

N = K - 1, and 0xXX is one byte of any value

26.3.1.4.6.4.5 BD Pointers Allocation

If the [ADJ] bit is cleared, **Table 26-23** describes the offset pointers allocation:

Table 26-23. BD Offset Pointers Assignment for Viterbi Direct Mode

Pointer	Viterbi Vector Rate 1/2	Viterbi Vector Rate 1/3	Viterbi Vector Rate 1/4
ADDR_OFFSET_0	Even [index] Interlaced P0 & P1 (#1)	Even [index] Interlaced P0, P1 & P2 (#1)	Even [index] Interlaced P0, P1, P2 & P3 (#1)
ADDR_OFFSET_1	Odd [index] Interlaced P0 & P1 (#2)	Odd [index] Interlaced P0, P1 & P2 (#2)	Odd [index] Interlaced P0, P1, P2 & P3 (#2)
ADDR_OFFSET_2	N/A	N/A	N/A
ADDR_OFFSET_3	N/A	N/A	N/A
ADDR_OFFSET_4	N/A	N/A	N/A
ADDR_OFFSET_5	N/A	N/A	N/A

If the [ADJ] bit is set, the MAPLE-B expects to find the first vector (vector #1) in system memory, in the address as described in the [BASE_ADDR] field of the BD. The second vector (vector #2) should begin in the first aligned-to-8-bytes address after the end of the first vector.

26.3.1.4.7 Separate Vectors Data Structure

In Separate Vectors Data Structure, the MAPLE-B TVPE expects to find the relevant input data types in separate vectors. The internal TVPE engine re-orders the input stream of data to fit its internal memory data structure, and it also generates interleaved systematic data (SD~) for the decoding process[1]. For the Separate Vectors Data Structure, the IN_D_STRCT field of the TVPE BD must be equal to 0001.

The supported vectors types for the 3GLTE and 3GPP operation modes (Binary decoding) are:

SD, PFa, PSa

The supported vectors types for the WiMAX operation mode (Duo-Binary decoding) are:

SDa, SDb, PFy, PFw, PSy, PSw.

The vectors structures as they should appear in system memory are illustrated in **Figure 26-4**.

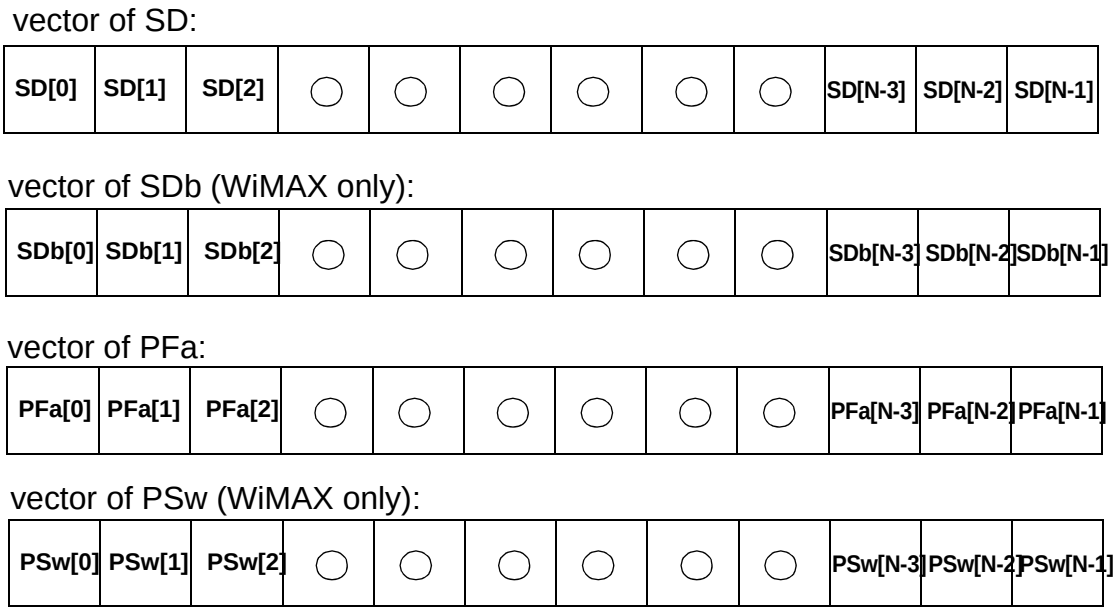


Figure 26-4. Separate Vectors Data Structure Example

Each data type element in the vector is in one byte resolution, and the total number of data type elements equals the current Block Size (BS field in TVPE BD).

The Separate Vectors input data structure can provide support to all supported Turbo operation modes.

1. 3GPP2 standard with large decoded Block Size requires special treatment as specified in **Section 26.3.1.4.11.1**, *3GPP2 Operation Mode—Large Blocks Treatment,* on page 26-52.

26.3.1.4.7.1 Zero Tail Identification for Separate Vectors

If Zero tail is enabled (BD[ZTTB]=1), the channel data stream includes trellis termination Tail bytes, which should be placed at the end of the input vectors. The Tail data structure in this mode must be configured by the host using the registers[1] listed in Section<Cross Refs> 26.4.4.1.4. Describing this set of registers as a table looks like **Table 26-24**.

Table 26-24. TVTTSIxCR Registers

TVTTSI0CR = SD	TVTTSI1CR = PFa	TVTTSI3CR = PSa
T_0	T_4	T_{12}
T_1	T_5	T_{13}
T_2	T_6	T_{14}
T_3	T_7	T_{15}
where T_x are the fields in the TVTTSI0CR, TVTTSI1CR and TVTTSI3CR registers (see **Table 26-24**), and are used to identify a specific Tail byte type (see **Table 26-112**). For the Separate Vectors Data Structure, each of the TVTTSI0CR, TVTTSI1CR, TVTTSI3CR registers is related to one data type (TVTTSI0CR=SD, TVTTSI1CR=PFa, TVTTSI3CR=PSa).		

The following example illustrates the use of the TVTTSIxCR registers using a case where the input vectors are of SD,PFa and PSa types. Therefore, according to **Table 26-24**, the fields to describe the Tail are T0-T3 for the SD vector (TVTTSI0CR), T4-T7 for the PFa vector (TVTTSI1CR) and T12 – T15 for the PSa vector (TVTTSI3CR).

Assume 3GLTE decoding operation mode, with Separate Vectors Data Structure. The expected vectors without the tail are:

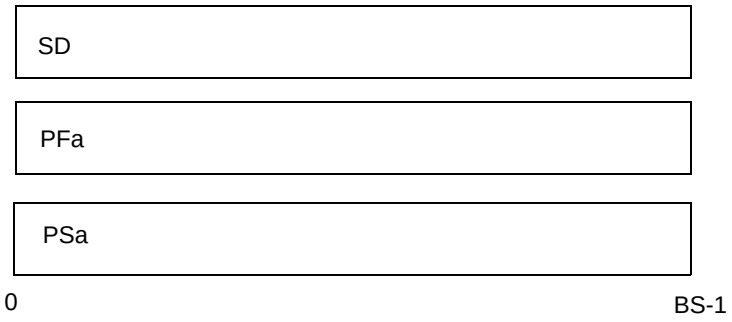
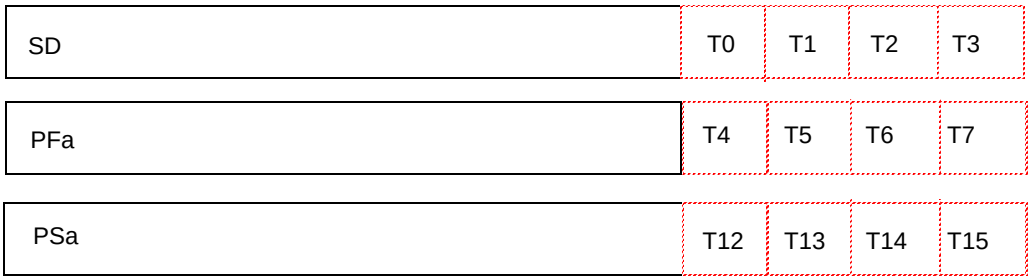


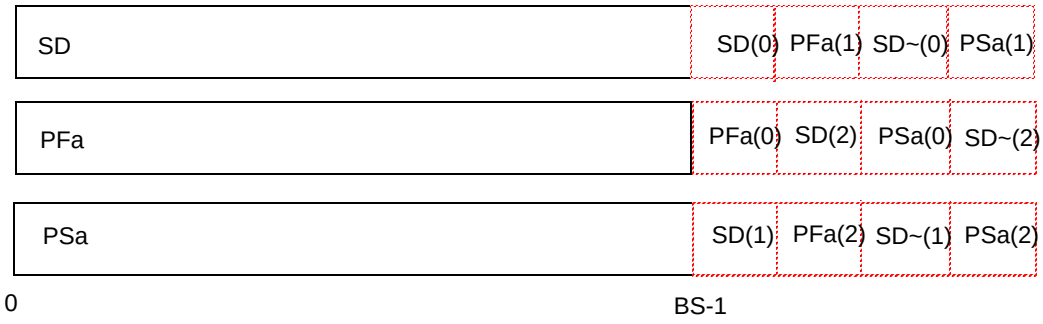
Figure 26-5. Input Vectors for 3GLTE with Separate Vectors Input Data Structure

1. The TVTTSIxCR registers are relevant only for Turbo decoding. For Viterbi processing each vector includes its own type of Tail bytes, therefore there is no need for a Tail description.

For rate 1/3 decoding, there are 12 bytes of Tail, which should be split between the 3 vectors (4 bytes to each vector). The Tail bytes, spread evenly across these 3 vectors in the order indicated in **Table 26-24**, are:



So, if the Tail bytes across the 3 vectors appear as:



then the registers should be programmed as follows:

Table 26-25. TVTTSIxCR Registers—Tail Configuration Example

TVTTSIxCR Field	TVTTSI0CR = SD	TVTTSI1CR = PFa	TVTTSI3CR = PSa
T0	SD(0)	PFa(0)	SD(1)
T1	PFa(1)	SD(2)	PFa(2)
T2	SD~(0)	PSa(0)	SD~(1)
T3	PSa(1)	SD~(2)	PSa(2)

The Tail byte types encoding is detailed in **Section 26.4.4.1.4**, *TVPE Turbo Tail Symbol Identification x Configuration Register (TVTTSIxCR)*. For example PFa(2) is encoded to 10 (001010).

26.3.1.4.7.2 BD Pointers Allocation for Separate Vectors Data Structure

If the [ADJ] bit in the TVPE BD is cleared, initialize the correct vectors for the standard to the BD offset pointers (ADDR_OFFSET_x) as indicated in **Table 26-26**.

Table 26-26. BD Offset Pointers Assignment Separate Vectors Data Structure

Operation Mode (Rate)	BS	TVPE BD Pointers					
		ADDR_OFFSET_0	ADDR_OFFSET_1	ADDR_OFFSET_2	ADDR_OFFSET_3	ADDR_OFFSET_4	ADDR_OFFSET_5
3GLTE 3GPP	N/A	SD	PFa	PSa	N/A	N/A	N/A
WiMAX	N/A	SDa	SDb	PFy	PFw	PSy	PSw

If the [ADJ] bit is set, the MAPLE-B expects to find the first vector in system memory, in the address as described in the [BASE_ADDR] field of the BD. The second vector should begin in the first aligned-to-8-bytes address at the end of the first vector, repeating this order until the last vector. For more details see **Section 26.3.1.4.12**, *Adjacent Bit Description ([ADJ])*, on page 26-54.

The order of the vectors, expected by MAPLE-B, is described in **Table 26-27**.

Table 26-27. Order of Separate Vectors in System Memory if [ADJ] Bit is Set

Operation Mode (Rate)	BS	Order of Vectors in Memory Staring From BASE_ADDR
3GLTE,3GPP	N/A	SD, PFa, PSa
WiMAX	N/A	SDa, SDb, PFy, PFw, PSy, PSw

26.3.1.4.8 Periodically Punctured Configurable Mix Stream (PPCMS) Data Structure

In PPCMS data structure, the MAPLE-B TVPE expects to find the relevant input data types in a single vector stream. The order of the symbols within the data stream is prescribed in **Table 26-28**.

For the PPCMS data structure, the IN_D_STRCT field of the TVPE BD must be equal to 0011.

Table 26-28. Default Symbol Ordering in PPCMS

Operation Mode	Rate	Default Symbol Ordering
All (Viterbi)	1/2	P0[n], P1[n], P0[n+1], P1[n+1], P0[n+2], P1[n+2], P0[n+3], P1[n+3]
All (Viterbi)	1/3	P0[n], P1[n], P2[n], P0[n+1], P1[n+1], P2[n+1]
All (Viterbi)	1/4	P0[n], P1[n], P2[n], P3[n], P0[n+1], P1[n+1], P2[n+1], P3[n+1]
3GPP2	1/5	SD[n], PFa[n], PFb[n], PSa[n], PSb[n], SD[n+1], PFa[n+1], PFb[n+1], PSa[n+1], PSb[n+1]

From **Table 26-27**: for Viterbi rate 1/2 processing, the expected input stream (after de-puncturing) is:

P0[0]	**P1[0]**	**P0[1]**	**P1[1]**	**P0[2]**	**P1[2]**	**P0[3]**	**P1[3]**	○	○	○	○

For Viterbi rate 1/3 processing, the expected input stream (after de-puncturing) is:

P0[0]	**P1[0]**	**P2[0]**	**P0[1]**	**P1[1]**	**P2[1]**	**P0[2]**	○	○	○	○	○

For Viterbi rate 1/4 processing, the expected input stream (after de-puncturing) is:

P0[0]	**P1[0]**	**P2[0]**	**P3[0]**	**P0[1]**	**P1[1]**	**P2[1]**	**P3[1]**	○	○	○	○

and for 3GPP2 operation mode, with Turbo rate 1/5, the expected input stream is:

SD[0]	**PFa[0]**	**PFb[0]**	**PSa[0]**	**PSb[0]**	**SD[1]**	**PFa[1]**	**PFb[1]**	**PSa[1]**	○	○	○

The MAPLE-B allows use of input streams with symbol order different than the default order as described in **Table 26-27**. Changing the symbols order within the stream is done using the TVSIxCR registers (**Section 26.4.4.1.2**, *TVPE Symbol Identification 0 Configuration Register (TVSI0CR)*).

The default values of the SIDx fields of the TVSIxCR registers (SID0=0, SID1=1, SID2=2, SID3=3, SID4=4,...,SID9=9) represent two[1] repetitions of the default symbols order for each possible input stream as described in **Table 26-27**. For example, for Viterbi rate 1/4, the SIDx default values represent the following:

SID0=0=P0[n]
SID1=1=P1[n]
SID2=2=P2[n]
SID3=3=P3[n]
SID4=4=P0[n+1]
SID5=5=P1[n+1]
SID6=6=P2[n+1]
SID7=7=P3[n+1]
SID8=8=0xXX
SID9=9=0xXX.

where n = 0, 2, 4...

1. For Viterbi rate 1/2 it represent four repetitions.

If the required input stream for the Viterbi rate 1/4 is not the default, but as follows:

P2[0]	**P1[0]**	**P3[0]**	**P0[0]**	**P2[1]**	**P1[1]**	**P3[1]**	**P0[1]**	**P2[2]**	**P1[2]**	○	○

then the required TVSIxCR[SIDx] configuration should be:

SID0=2=P2[n]
SID1=1=P1[n]
SID2=3=P3[n]
SID3=0=P0[n]
SID4=6=P2[n+1]
SID5=5=P1[n+1]
SID6=7=P3[n+1]
SID7=4=P0[n+1]
SID8=0xXX=XX
SID9=0xXX=XX

Note: The PPCMS data structure can support a periodic punctured input data stream, but for the TVSIxCR fields configuration, the input stream must be described as it should appear after the de-puncturing (or before the puncturing process).

When working in PPCMS, besides the BS field which indicates the decoded block size, the BUF_SIZE field must be initialized, as well, with the actual size of the data stream in bytes. The value of BUF_SIZE should be calculated according top the following:

```
if ([ZTTB]=1)
    BUF_SIZE = roundup{{([BS] + (K - 1) + 7)&0xFFF8}*EncRate*PuncRate}
else
    BUF_SIZE = roundup{{([BS] + 7)&0xFFF8}*EncRate*PuncRate}
```

where:

[ZTTB], [BS] -- Fields from the TVPE BD.

K -- Viterbi Constraint Length.

EncRate -- Encoder rate. Should be equal 2, 3 or 4 for encoder rates of 1/2, 1/3 or 1/4 respectively.

PuncRate -- Puncturing rate. For example 2/3, 3/4 etc.

For example: If [BS]= 644, [ZTTB] =1, K=9, Encoding rate of 1/3 and Puncturing rate of 2/3:

BUF_SIZE = {(644 + (9-1) +7)&0xFFF8}*3*2/3 = {659&0xFFF8}*2 = 656*2=1312

The PPCMS input data structure supports a periodically punctured input data stream, and the MAPLE-B is capable of performing “on-the-fly” periodic de-puncturing of the input stream. The MAPLE-B uses the MTVPVx(H/L)CP and MTVPPCyP parameters (see **Section 26.4.2.2.2**, *MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter (MTVPVxHCP)*, on page 26-154, **Section 26.4.2.2.3**, *MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter (MTVPVxLCP)*, on page 26-155, and **Section 26.4.2.2.4**, *MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter (MTVPPCyP)*, on page 26-156), with the PUNC_SCHM field of the TVPE BD to configure the de-puncturing scheme of the input stream. The MAPLE-B can be configured to hold up to 10 different sets of puncturing vectors (63 bits) and puncturing period parameters. The PUNC_SCHM field in the TVPE BD chooses the correct set for the current job. If the PUNC_SCHM equals 0000 then no de-puncturing is executed.

Each puncturing set includes two elements:

- Puncturing vector—63 bit vector which describes where in the input stream de-puncturing is expected. This is done by clearing the relevant bits in the vector.
- Puncturing period parameter—describes how many bits in the puncturing vector are valid.

The following pseudo-code describes the de-puncturing algorithm:

```
j=0; /* de-punctured output block pointer */

i=0; /* punctured input block pointer */

foreach byte in the input buffer

 if PVEC[i%PPER] = 1 then depunctured_block[j] = input_block[i] and increment i & j;

 if PVEC[i%PPER] = 0 then depunctured_block[j] = 0 and j is increment;

where PVEC, the puncturing vector, and PPER, the puncturing period, are located in
the MTVPVx(H/L)CP and MTVPPCyP fields, respectively.
```

Figure 26-6 describes an example of the Viterbi decoding periodic de-puncturing of a rate 1/3 encoder:

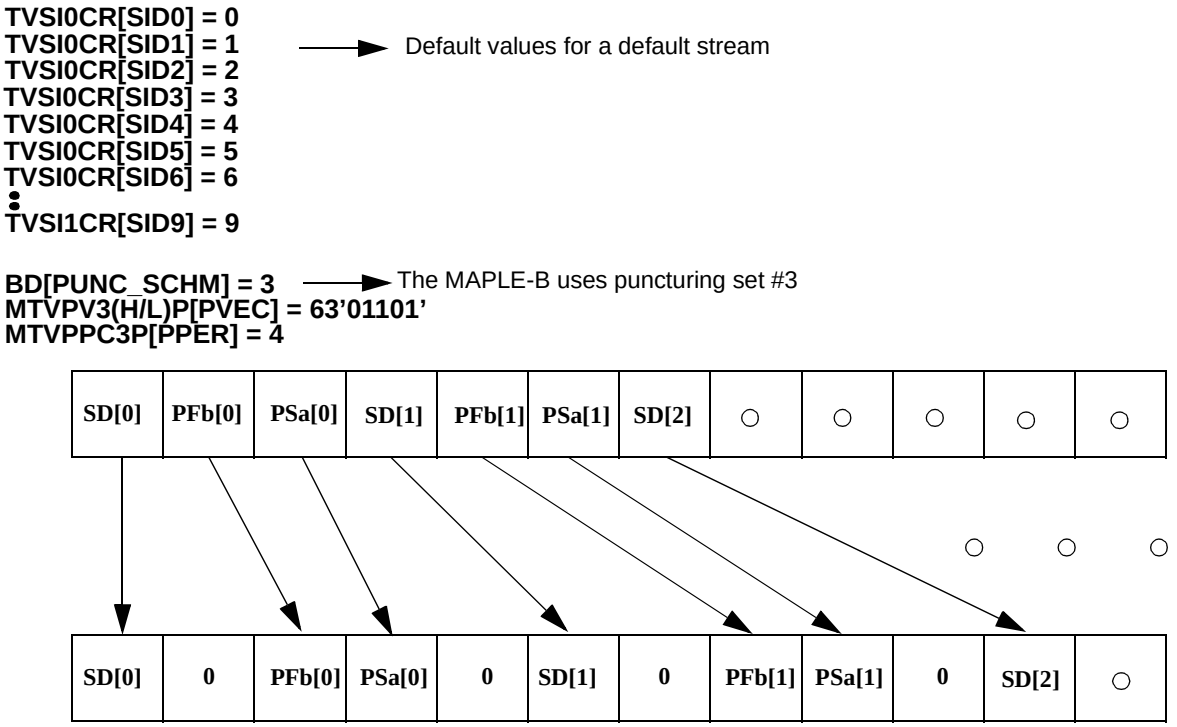


Figure 26-6. De-Puncturing Process of PPCMS Input Data Structure

Note: When [PPER] is set to 1 and [PVEC] is set to 0x0000_0000_0000_0001, the result is a non-punctured periodic mixed stream of symbols.

26.3.1.4.8.1 Zero Tail Identification for PPCMS

For Viterbi decoding, the Tail bytes are expected at the end of the vector with the same order and with the same puncturing scheme as the "body" bytes. For example, for Viterbi rate 1/3, if the order of the "body" bytes, as defined in TVSIxCR[SIDx], is: *P1[x], P0[x], P2[x],...* The expected order of the Tail bytes at the end of the vector then are: $P1_t[0]$, $P0_t[0]$, $P2_t[0]$,..., $P1_t[K-2]$, $P0_t[K-2]$, $P2_t[K-2]$.

where K is the Viterbi Constraint length.

26.3.1.4.8.2 BD Pointers Allocation for PPCMS

Because the PPCMS input data structure includes only one input vector, the MAPLE-B expects to find the pointer to system memory for that vector in the BASE_ADDR field of the BD, regardless of the [ADJ] bit value.

26.3.1.4.9 Rate-Matched Fixed Mix Stream Data Structure

For this input data structure, the TVPE executes Rate-De-Matching using de-puncturing or using a de-repetition scheme. This scheme provides support for E-DCH, as defined in 3GPP TS 25.212 Multiplexing and channel coding, section 4.8.

For the Rate-Matched Fixed Mix Stream data structure, the IN_D_STRCT field of the TVPE BD must be equal to 0110.

The Rate-Matched Fixed Mix Stream includes one input vector which includes a mix of all the data types as defined in the standard mentioned above.

Since the 3GPP TS 25.212 standard defines the rate-matching process over transport block (TB) and not code block (CB) and since the TVPE works on CB only, the following parameters in the TVPE BD must be initialized:

[BUF_SIZE] - When working in 'Rate-Matched Fixed Mix Stream', besides the BS field which indicates the decoded block size, the BUF_SIZE field must be initialized, as well, with the actual size of the data stream in bytes. The value of the BUF_SIZE should reflect the actual size of the input data buffer to be input into the TVPE.

[E_PARAM_PTR] - A pointer to a data structure in the system memory which includes the following parameters[1]:

e_cb_sd_init[2] - The e_{init} parameter with the additional offset of the current code block in the transport block for the systematic data

e_sd_plus - The e_{plus} parameter for the systematic data

e_sd_minus - The e_{minus} parameter for the systematic data

e_cb_pfa_init[3] - The e_{init} parameter with the additional offset of the current code block in the transport block for the first parity data (p1).

e_pfa_plus - The e_{plus} parameter for the first parity data (p1)

e_pfa_minus - The e_{minus} parameter for the first parity data (p1)

p_cb_psa_init[4] - The e_{init} parameter with the additional offset of the current code block in the transport block for the second parity data (p2).

e_psa_plus - The e_{plus} parameter for the second parity data (p2)

e_psa_minus - The e_{minus} parameter for the second parity data (p2)

1. All the parameters definition is taken from the 3GPP TS 25.212, section 4.5.4.3 "HARQ Second Rate Matching Stage" standard.
2. Calculating this parameter requires additional calculation over the e_{init} parameter as defined in the 3GPP TS 25.212, section 4.5.4.3 "HARQ Second Rate Matching Stage" standard.
3. Calculating this parameter requires additional calculation over the e_{init} parameter as defined in the 3GPP TS 25.212, section 4.5.4.3 "HARQ Second Rate Matching Stage" standard.
4. Calculating this parameter requires additional calculation over the e_{init} parameter as defined in the 3GPP TS 25.212, section 4.5.4.3 "HARQ Second Rate Matching Stage" standard.

Each of the above parameters must be represented by 15 bits representation and the whole data structure pointed by the [E_PARAM_PTR] should be arranged as described in **Figure 26-7**:

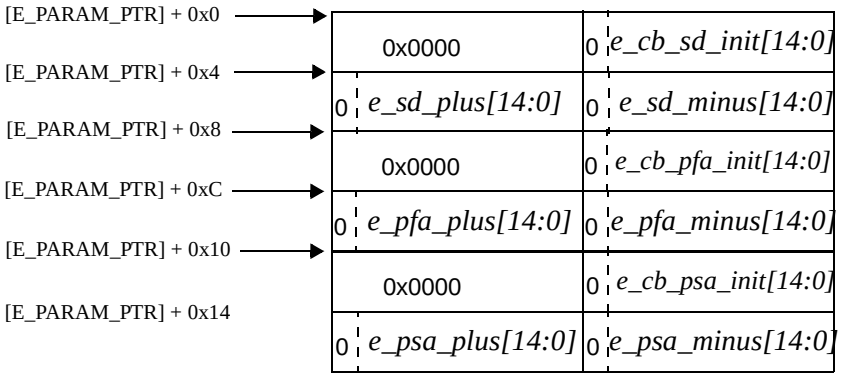


Figure 26-7. E parameters data structure description

[DREP] - Indication to the TVPE whether de-puncturing scheme is required ([DREP]=0) or de-repetition ([DREP]=1).

The de-puncturing process is similar to the one described in figure **Figure 26-6**, with the single difference that the regularity of the repetition is non-coherent and behaves according to the pseudo code described in **Section 26.3.1.4.9.1**, *Rate De-Matching Using De-Puncturing*.

Figure 26-8 describes an example of a possible de-repetition process:

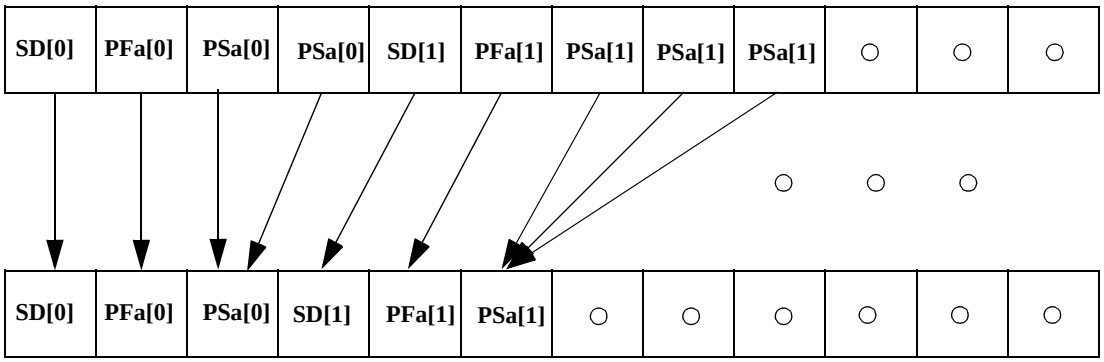


Figure 26-8. De-rEpetition Process Example

The regularity of the repetition is not provided in the example, as it is non-coherent. It is provided in the **Section 26.3.1.4.9.2**, *Rate De-Matching Using De-Repetition*, on page 26-47 pseudo code in a subsequent section.

26.3.1.4.9.1 Rate De-Matching Using De-Puncturing

The following pseudo code describes the process of Rate De-Matching using de-puncturing on the input data:

```
m = 0;
processed_stream_pointer = 0;
// When writing to the stream processor
switch (processed_stream_pointer% 3) {
```

```
// Systematic bit
case 0: {
 e_cb_sd_init = e_cb_sd_init - e_sd_minus;
 if (e_cb_sd_init <= 0) {
     processed_stream[processed_stream_pointer++] = 0;
     e_cb_sd_init = e_cb_sd_init + e_sd_plus;
}
 else {
      processed_stream[processed_stream_pointer++] = input_stream[m++];
}
 break;
}

case 1: {
 e_cb_pfa_init = e_cb_pfa_init - e_pfa_minus;
 if (e_cb_pfa_init <= 0) {
     processed_stream[processed_stream_pointer++] = 0;
     e_cb_pfa_init = e_cb_pfa_init + e_pfa_plus;
}
 else {
      processed_stream[processed_stream_pointer++] = input_stream[m++];
}
 break;
}

case 2: {
 e_cb_psa_init = e_cb_psa_init - e_psa_minus;
 if (e_cb_psa_init <= 0) {
     processed_stream[processed_stream_pointer++] = 0;
     e_cb_psa_init = e_cb_psa_init + e_psa_plus;
}
 else {
      processed_stream[processed_stream_pointer++] = input_stream[m++];
}
 break;
}
}
```

where e_{plus}, e_{minus}, and e_{init} of SD,PFa and PSa are taken from the data structure pointed to by the [E_PARAM_PTR] pointer in TVPE BD.

26.3.1.4.9.2 Rate De-Matching Using De-Repetition

The following pseudo code describes the process of Rate De-Matching using de-repetition on the input data:

```
m = 0;
processed_stream_pointer = 0;
state_sd = NORMAL;
state_pfa = NORMAL;
state_psa = NORMAL;
// When writing to the channel data memory
switch (processed_stream_pointer% 3) {

// Systematic bit
case 0: {
 if (state_sd == REPETITION) {
     m++;
     e_cb_sd_init = e_cb_sd_init + e_sd_plus;
     if (e_cb_sd_init <= 0)
        state_sd = REPETITION;
     else {
        state_sd = NORMAL;
        processed_stream_pointer++;
}
}
 else {
     processed_stream[processed_stream_pointer] = input_stream[m++];
     e_cb_sd_init = e_cb_sd_init - e_sd_minus;
     if (e_cb_sd_init <= 0) {
        state_sd = REPETITION;
}
     else {
        state_sd = NORMAL;
        processed_stream_pointer++;
}
}
 break;
} /* of switch 0 */
// PFa bit

case 1: {
 if (state_sd == REPETITION) {
     m++;
     e_cb_pfa_init = e_cb_pfa_init + e_pfa_plus;
     if (e_cb_pfa_init <= 0)
        state_sd = REPETITION;
     else {
        state_sd = NORMAL;
        processed_stream_pointer++;
}
}
 else {
     processed_stream[processed_stream_pointer] = input_stream[m++];
     e_cb_pfa_init = e_cb_pfa_init - e_pfa_minus;
     if (e_cb_pfa_init <= 0) {
        state_sd = REPETITION;
}
     else {
        state_sd = NORMAL;
```

```
        processed_stream_pointer++;
}
}
 break;
} /* of switch 1 */

// PSa bit
case 2: {
 if (state_sd == REPETITION) {
     m++;
     e_cb_psa_init = e_cb_psa_init + e_psa_plus;
     if (e_cb_psa_init <= 0)
        state_sd = REPETITION;
     else {
        state_sd = NORMAL;
        processed_stream_pointer++;
}
}
 else {
     processed_stream[processed_stream_pointer] = input_stream[m++];
     e_cb_psa_init = e_cb_psa_init - e_psa_minus;
     if (e_psa_init <= 0) {
        state_sd = REPETITION;
}
     else {
        state_sd = NORMAL;
        processed_stream_pointer++;
}
}
 break;
} /* of switch 2 */
} /* of switch */
```

where the e_{plus}, e_{minus}, and e_cb_{init} of SD, PFa, and PSa are taken from the data structure pointed to by the [E_PARAM_PTR] pointer in the TVPE BD.

26.3.1.4.9.3 Zero Tail Identification for Rate-Matched Fixed Mix Stream

In this input data structure, there is no need for special treatment for the tail because the tail structure is well defined in the standard, and the MAPLE-B always includes the tail in its processing.

26.3.1.4.9.4 BD Pointers Allocation for Rate-Matched Fixed Mix Stream

Because the Rate-Matched Fixed Mix Stream input data structure includes only one vector, the MAPLE-B expects to find the pointer to system memory for that vector in the BASE_ADDR field of the BD regardless of the [ADJ] bit value. For details see **Section 26.3.1.4.12**, *Adjacent Bit Description ([ADJ])*, on page 26-54.”

26.3.1.4.10 Sub-Block Interleaved Vectors Data Structure

The Sub-Block Interleaved Vectors input data structure provides support for Turbo decoding, as it appears in the WiMAX OFDMA turbo decoding (**IEEE** 802.16-2004 standard and corrigendum **IEEE** P802.16-2004/Cor2/D2, section 8.4.9.2), and the 3GLTE (Evolved UTRA) sub-block de-interleaving and rate matching selected proposal (CBRM), and therefore is valid only during 3GLTE & WiMAX[1] operation modes.

For the Sub-Block Interleaved Vectors data structure, the IN_D_STRCT field of the TVPE BD must be equal to 0111.

For this input data structure, the TVPE executes on-the-fly sub-block de-interleaving on the sub-block interleaved input data vectors.

There are two types of sub-block de-interleaving that can be executed by the TVPE:

- Singles sub-block de-interleaving. The TVPE performs on-the-fly sub-block de-interleaving on one type of data, that is, on SD only (SDa only and SDb only for WiMAX). **Figure 26-9** describes an example of single sub-block de-interleaving:

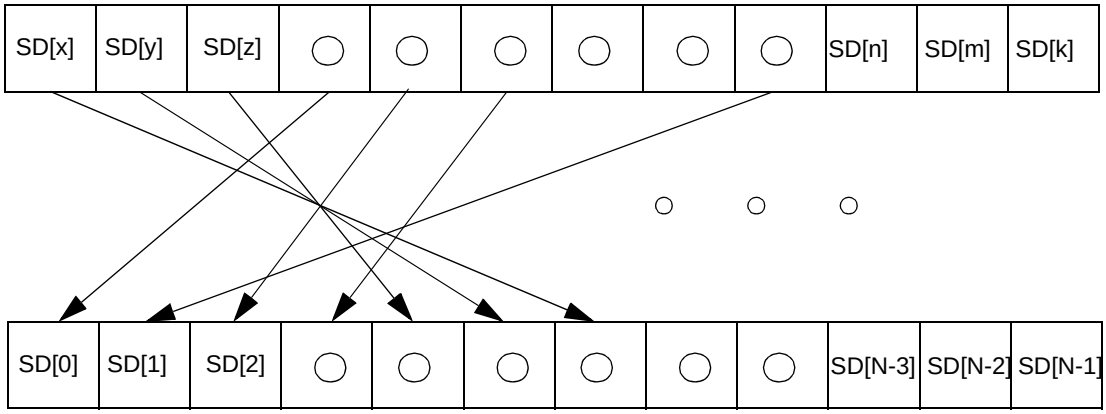


Figure 26-9. Single Sub Block De-Interleaving Example

- Pairs sub-block de-interleaving. The TVPE performs on-the-fly sub-block de-interleaving on pairs of data types, that is, on PFa+PSb, PFy+PSy and PFw+PSw. **Figure 26-9** describes an example of pairs sub-block de-interleaving:

1. The ‘Sub-Block Interleaved Vectors’ is not supported for the WiMAX IEEE® 802.16m-09/0010r2 standard.

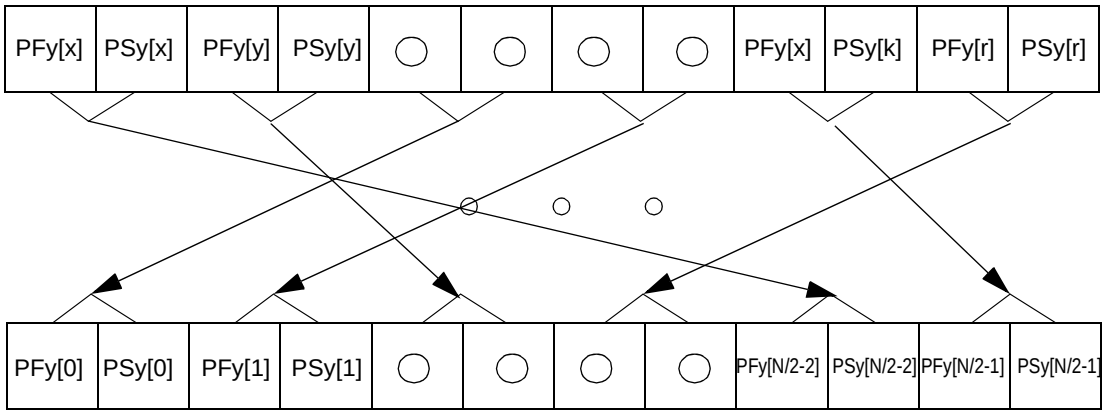


Figure 26-10. Pairs Sub Block De-Interleaving Example

Table 26-29 describes the type of sub-block de-interleaving executed for each of the possible operation modes:

Table 26-29. Sub Block De-Interleaving With Respect to Operation Mode

Operation Mode	Single Sub-Block De-Interleaving	Pairs Sub-Block De-Interleaving
3GLTE	SD	PFa+PSa
WiMAX	SDa SDb	PFy+PSy PFw+PSw

The expected input vector for 3GLTE is described as follows:

- For the 3GLTE operation mode, the MAPLE-B expects one input vector which should point to the system level HARQ buffer (CBRM). This buffer should be maintained by the external master and should include the systematic and parity information for the decoding.
- The HARQ buffer includes the sub-block interleaved systematic data and the sub block interleaved and interlaced PFa+PSa data. The MAPLE-B internally calculates the sizes of the SD buffer and the PFa+PSa buffer, and the TVPE performs the sub-block de-interleaving accordingly.

The expected input vector for WiMAX is described as follows:

- For the WiMAX operation mode, the MAPLE-B expects a pointer to the following four input vectors:
  - Sub-block interleaved SDa vector.
  - Sub-block interleaved SDb vector.
  - Sub block interleaved PFy+PSy pairs.
  - Sub block interleaved PFw+PSw pairs.
- The TVPE performs sub-block interleaving for each of the blocks internally and place the results in its internal memories.

26.3.1.4.10.1 Zero Tail identification for Sub-Block Interleaved Vectors

The WiMAX standard assumes Tail biting trellis termination. Therefore there is no need for Tail byte handling.

In the 3GLTE standard, the Tail bytes are sub-block interleaved with the data bytes. After the sub-block de-interleaving is executed by the TVPE, the Tail bytes should be located at the end of the SD vector, and the interlaced PFa+PSa vector. The order of the Tail byte types at the end of the vectors is self configured by the MAPLE-B according to the order derived from the 3GLTE Evolved UTRA.

26.3.1.4.10.2 BD Pointers Allocation for Sub-Block Interleaved Vectors Data Structure

Because for the 3GLTE operation mode and Sub-Block Interleaved Vectors data structure there is only one input vector (system HARQ buffer), the MAPLE-B expects to find it in the BASE_ADDR field of the TVPE BD, regardless of the [ADJ] bit value.

For the WiMAX operation mode, if the [ADJ] bit is cleared, the MAPLE-B expects to find four input vectors which are to be assigned to the [ADDR_OFFSET_x] fields of the TVPE BD according to **Table 26-30**.

Table 26-30. BD Offset Pointers Assignment Sub-Block Interleaved Vectors Data Structure

	TVPE BD Pointers					
Operation Mode (Standard)	**ADDR_OFFSET_0**	**ADDR_OFFSET_1**	**ADDR_OFFSET_2**	**ADDR_OFFSET_3**	**ADDR_OFFSET_4**	**ADDR_OFFSET_5**
WiMAX	sub-block interleaved SDa	sub-block interleaved SDb	sub-block interleaved PFy + PSy	sub-block interleaved PFw + PSw	N/A	N/A

If the [ADJ] bit is set, the MAPLE-B expects to find the first vector in system memory, in the address described in the [BASE_ADDR] field of the BD. The second vector should begin in the first aligned-to-8-bytes address after the end of the first vector, repeating this order until the last vector. For details, see **Section 26.3.1.4.12**, *Adjacent Bit Description ([ADJ])*, on page 26-54.

The order of the vectors in the system expected by the MAPLE-B is:

For WiMAX: SDa, SDb, PFy+PSy, PFw+PSw.

26.3.1.4.11 Supported Input Data Structures for Different Standards

Table 26-31 summarizes the possible input data structures for MAPLE-B operation modes:

Table 26-31. Supported Input Data Structure with Respect to Configured Standard

Operation Mode (Standard)	Input Data Structure				
	Direct	Separate Vectors	Periodically Punctured Configurable Mix Stream	Rate-Matched Fixed Mix Stream	Sub Block Interleaved Vectors
3GLTE	Yes	Yes	No	No	Yes
WiMAX	Yes	Yes	No	No	Yes[1]
3GPP	Yes	Yes	No	Yes	No
3GPP2	Yes	No	No	No	No
Viterbi[2]	Yes	No	Yes	No	No

1. .The 'Sub-Block Interleaved Vectors' is not supported for the WiMAX IEEE® 802.16m-09/0010r2 standard.
2. Viterbi decoding does not require a special operation mode of the MAPLE-B. It is fully programmable regardless of the MAPLE-B operating mode.

26.3.1.4.11.1 3GPP2 Operation Mode—Large Blocks Treatment

While working on blocks whose size does not exceed the TVPE internal memory space, the TVPE executes internally until either the number of iterations as described in the TVPE BD field [MAX_ITER] is complete or until Stop Criteria is activated (see **Section 26.3.2.1.1.5**, *Stopping Criteria*).

The MAPLE-B also supports Turbo decoding for blocks larger than the available internal TVPE memory space. The following defines the Large Block Size (LBS) indicating the 3GPP2 Block size values that exceed the TVPE internal memory space.

If (BS > 4K), BS is treated as LSB

To execute these large blocks, the MAPLE-B uses a partitioning scheme called striping. During this scheme, the MPALE-B splits the decoded block into stripes and works on each stripe separately. For each half iteration processing, the MAPLE-B internally manages fetching the different stripes.

When Turbo decoding a BS that meets the LBS definition, the only supported input data structure is the Direct input data structure. This operation mode requires no additional configuration from the host.

The MAPLE-B assumes striping mode for 3GPP2 only and for Block Sizes that meet the LBS.

Note: When the MAPLE-B uses the striping mode to decode LBS, the data transfer on the MAPLE-B external interface increases significantly.

26.3.1.4.11.2 3GLTE Supported Block Sizes (BS)

The 3GLTE CBRM allows transmission of any code block size ranging from 1 to 6144 bits. Part of the CBRM generation in the transmitter is to turn the code block size into QPP[1] size by adding filler cells before the sub-block interleaving process. Since the MAPLE-B Sub-Block Interleaved input data structure include sub-block de-interleaving process, it can support any block size, and is not limited to QPP sizes only.

For the 3GLTE Separate Vectors or Direct input data structures, the input data is already included after sub-block de-interleaving processing; therefore, the MAPLE-B can also support block sizes that are only of QPP sizes. If the transmitted code block is not of QPP size, the host performs the external sub-block de-interleaving to add the required number of filler bits at the beginning of the SD vector and the PF vector. Filler bits are added to reach equal vector length with the PS vector, which already includes embedded filler cells. Note that each decoded bit is represented using a 2's complement representation byte. The added bits must use the logic 0 value, which its representation depends on the 'mode' parameter initialization See the API (see *Initialization Parameters Structure* in the **API User's Manual**).

Table 26-32 summarizes the possible BS values for the different input data structures in 3GLTE operation mode:

Table 26-32. Possible BS values with respect to input data structure for 3GLTE

Input Data Structure	Possible BS Values
'Direct'	Only 188 QPP sizes ranging from 40 to 6144
'Separate Vectors'	Only 188 QPP sizes ranging from 40 to 6144
'Sub-Block Interleaved'	All sizes ranging from 1 to 6144

26.3.1.4.11.3 Expected Vectors Example for 3GLTE Separate Vectors

Assume a transmitted code block with 171 bits length (see *Padding* in the **API User's Manual** for details). After sub-block de-interleaving and byte separation according to their type, there are the following three vectors with the following lengths:

- SD vector. Length 171 + 4.
- PF vector. Length 171 + 4.
- PS vector. Length 171 + 4 + 5 (the +5 are the embedded filler cells added during the encoding stage in the transmitter.

1. QPP size - Valid 3GLTE interleaver size out the 188 possible sizes as defined in Table 5.1.3.3 of the 3GPP TS 36.212 document.

Because the MAPLE-B expects equal vectors lengths, the host must pad the beginning of the SD and PF vectors with 5 logic 0 values for each vector, as shown in **Figure 26-11**.

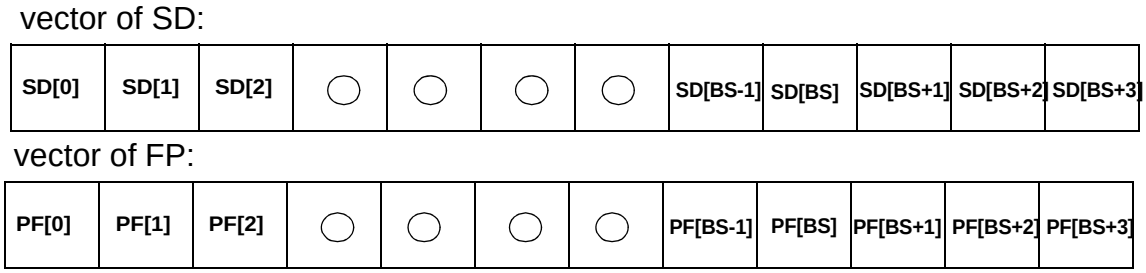


Figure 26-11. Padding Vectors SD and FP

26.3.1.4.12 Adjacent Bit Description ([ADJ])

As previously mentioned in the input data structures sections, the [ADJ] bit in the TVPE BD allows locating all the relevant input vectors successively, thus saving the need to initialize specific offset addresses for each of the vectors. **Figure 26-12** describes how the data structure should be placed in system memory (128 bit wide example) with respect to the [ADJ] bit. The example describes possible system memory content for the 'Separate Vectors' data structure for WiMAX operation mode (valid vectors: SDa, SDb, PFy, PFw, PSy, PSw).

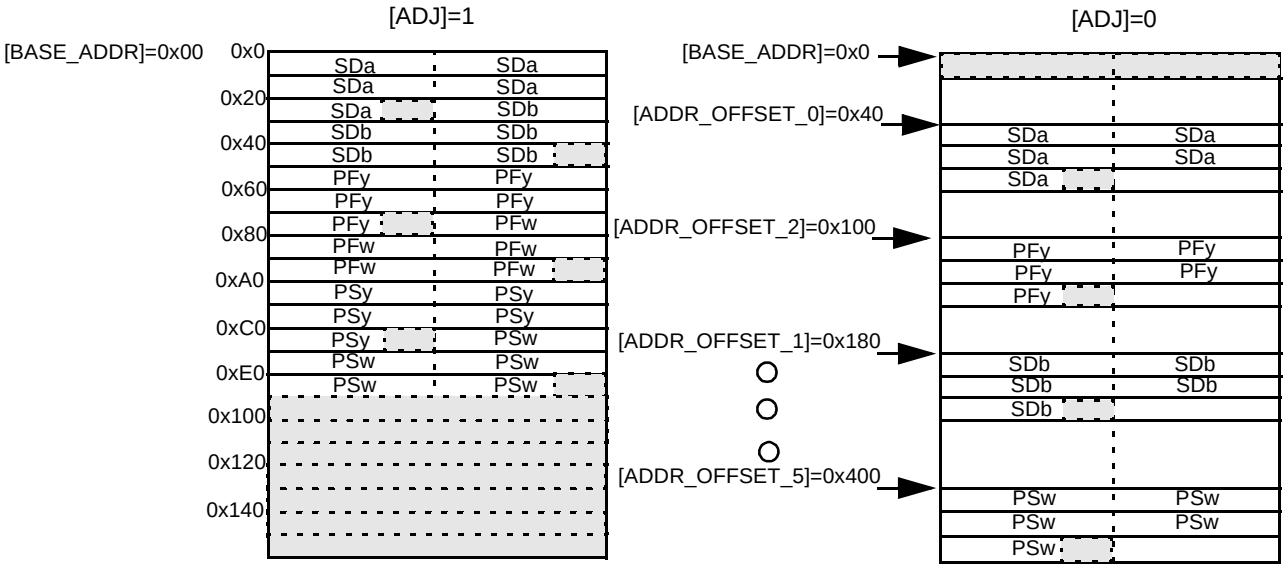


Figure 26-12. Possible System Memory Content for WiMAX 'Separate Vectors' Data Structure

When [ADJ]=0, the offsets need to be 64-Byte aligned, as the [ADDR_OFFSET_x] field represents the upper 16 bits of a 22-bit offset address.

26.3.2 PE Operations

The following sections provide details on the processing operations for the following PEs:

- Turbo/Viterbi PE
- FFT/iFFT and DFT/iDFT PEs
- CRC PE

26.3.2.1 Turbo/Viterbi PE

This section describes the operation performed by the Turbo/Viterbi PE. The sections discuss the following areas:

- Turbo decoding processing
- Viterbi decoding processing
- TVPE output data
- TVPE debug and profiling

26.3.2.1.1 Turbo Decoding Processing

The MAPLE-B supports four Turbo decoding standards in its possible operation modes:

- 3GLTE standard operation mode
- WiMAX standard operation mode
- 3GPP standard operation mode
- 3GPP2 standard operation mode.

Once the MAPLE-B operation mode is set during initialization (see **Section 26.3.4**), it can process Turbo decoding only according to its initialized operation mode.

26.3.2.1.1.1 Turbo Channel Data Interleaving

Before the TVPE can start its decoding operation, the interleaved data channel must be generated. The MAPLE-B uses its internal parameters/registers, which have being set during the MAPLE-B operation mode settings, and the BS (Block Size) field of the TVPE BD (**Section 26.3.1.4.2**, *TVPE Buffer-Descriptor Structure*) to internally determine the interleaving configuration. The interleaving process is executed internally and independently by the MAPLE-B TVPE[1].

1. For Direct Data Structure (**Section 26.3.1.4.6**, *Direct Data Structure*) or 3GPP2 large blocks (**Section 26.3.1.4.11.1**, *3GPP2 Operation Mode—Large Blocks Treatment*), the interleaved data channel must be generated externally to the MAPLE-B.

26.3.2.1.1.2 Number of Turbo Processing Elements (DREs)

The TVPE includes four DRE (Dual Recursion Element) engines, which are used for Turbo processing. For different block sizes, it is possible to activate either 1 DRE engine ([NDRE]=’01’), 2 DRE engines ([NDRE]=’10’) or 4 DRE engines ([NDRE]=’11’). When activating more than one such engine during a given Turbo processing session, the TVPE internally divides the block into internal sub-blocks for each of the active DREs, thus paralleling the decoding process and increasing the decoding speed of the TVPE. The MAPLE-B, in its default configuration ([NDRE]=00), is programed to work with the maximum possible number of DREs per given block size in order to achieve maximum performance. **Table 26-33** describes the number of DREs used by the TVPE in the MAPLE-B’s default configuration. The maximum number of DREs depends on the operation mode and the Block Size (BS) value.

Table 26-33. Maximum possible number of DREs

Operation Mode	Maximum number of DREs

3GLTE	if (BS ≥ 64)
	4 DREs
	else
	2 DREs

WiMAX	if (BS ≥ 96)
	4 DREs
	else
	2 DREs

3GPP,3GPP2	if (BS ≥ 128)
	4 DREs
	else
	2 DREs

Note: Working with a number of DRE engines which exceed the limitations described in **Table 26-33** may result in an unknown state.

Note: Working with less DRE engines than described in **Table 26-33** results in fewer internally divided sub-blocks of the decoded block and may achieve slightly better BLER performance. For details on the affect of the number of active DREs on the BLER performance, see the API User’s Manual.

Activating less DRE engines (with respect to **Table 26-33**) for a given block size should be done with the following limitations:

Table 26-34. Minimum DREs number with respect to Block Size

Total Block Size[1]	Minimum number of DREs ([NDRE] value)
Total Block Size ≤ 1024	1 (01)
1024 < Total Block Size ≤ 2048	2 (10)
2048 < Total Block Size	4 (11)

1. The Total Block Size as defined in this table should include the 3 Tail bits (if exist), that is, “Total Block Size” = [BS] + 3.

Setting the number of DREs is done using the [NDRE] field of the TVPE BD and according to the following decoding:

- NDRE[1:0]=00 -- Number of DREs is set by MAPLE-B in order to achieve maximum throughput performance.
- NDRE[1:0]=01 -- One DRE is active.
- NDRE[1:0]=10 -- Two DREs are active.
- NDRE[1:0]=11 -- Four DREs are active.

26.3.2.1.1.3 Turbo Processing Implementation

The following equations describe the TVPE Turbo implementation with respect to the possible LogMAP methods. The Radix-4 implementation is implemented for Binary decoding only.

Gamma Calculation:

$$\gamma_k(s',s) = \frac{SD_n \cdot esd_n + SD_{n+1} \cdot esd_{n+1} + P_n \cdot ep_n + P_{n+1} \cdot ep_{n+1} + \frac{3}{4} \cdot (EXT_n \cdot esd_n + EXT_{n+1} \cdot esd_{n+1})}{2}$$

where SD_n, SD_{n+1}, P_n & P_{n+1} are the 8 bit received systematic and parity channel data for bit n and n+1,

and Ext_n & Ext_{n+1} are the extrinsic values from the previous iteration (0 on the first iteration),

and esd_n, esd_{n+1}, epn, ep_{n+1} are the expected values for transition between states s‘ and s.

Alpha Recursion:

$$\alpha_k(s) = ACS4_{s' \subseteq S}(\alpha_{k-1}(s') + \gamma_k(s', s))$$

Beta Recursion:

$$\beta_k(s) = ACS4_{s' \subseteq S}(\beta_{k+1}(s') + \gamma_{k+1}(s', s))$$

Lambda Calculation:

$$\Lambda_k(u, v) = ACS8_{s' \subseteq S}(\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s))$$

where u & v are the inputs that cause a transition between state s‘ and state s.

Hard Outputs are h_n and h_{n+1} such that:

$$\Lambda(h_n,h_{n+1}) = max(((\Lambda_k(0, 0), \Lambda_k(0, 1)), \Lambda_k(1, 0)), \Lambda_k(1, 1))$$

ACS2 definition when using MaxLogMap is:

$$ACS2M = cmax(u, v)$$

where cmax is the cyclical max between u & v:

ACS2 definition when using LinearLogMap is:

$$ACS2L = cmax(u, v) + f(u, v, lcf)$$

where lcf is the [LLMAP_LCF] field in the TVPE BD and f is defined as:

$$f(u, v, lcf) = \begin{cases} \frac{lcf}{2} - \frac{|u - v|}{2}, & |u - v| < lcf \\ 0, & Otherwise \end{cases}$$

ACS4 in the case of LinearLogMap is defined as:

$$ACS4L(u, v, w, y) = ACS2M((ACS2L[u, v]), ACS2L[w, y])$$

and in the case of MaxLogMap as:

$$ACS4M(u, v, w, y) = ACS2M((ACS2M[u, v]), ACS2M[w, y])$$

ACS 8 in the case of LinearLogMap is defined as:

$$ACS8L(a, b, c, d, e, f, g, h) = ACS2M((ACS4L[a, b, c, d]), ACS4L[e, f, g, h])$$

and in the case of MaxLogMap as:

$$ACS8M(a, b, c, d, e, f, g, h) = ACS2M((ACS4M[a, b, c, d]), ACS4M[e, f, g, h])$$

26.3.2.1.1.4 CRC Check

The MAPLE-B can execute a CRC check on the Hard Outputs of the Turbo decoded blocks. The CRC check, which is executed internally by a dedicated hardware engine (CRCPE), supports four possible CRC polynomials:

1. CRC24 with polynomial $D^{24} + D^{23} + D^6 + D^5 + D + 1$
2. CRC24 with polynomial $D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$
3. CRC16-CCITT with polynomial $D^{16} + D^{12} + D^5 + 1$
4. CRC16 with polynomial $D^{16} + D^{15} + D^{14} + D^{11} + D^6 + D^5 + D^2 + D + 1$

Choosing a polynomial using the [CRC_POLY] field of the TVPE BD also sets the following configurations for the CRCPE parameters as described in **Section 26.3.2.3.1**, *CRC Buffer-Descriptor Structure*:

Table 26-35. CRCPE Configuration set determined by [CRC_POLY] of the TVPE BD

[CRC_POLY] of the TVPE BD	CRCPE Configuration
[CRC_POLY] = 00 01 11	CRCPE[RVRS_IN] = 1
	CRCPE[RVRS_OUT] = 1
	CRCPE[INV_OUT] = 0
	CRCPE[CRC_INIT] = 0x000000
[CRC_POLY] = 10	CRCPE[RVRS_IN] = 1
	CRCPE[RVRS_OUT] = 1
	CRCPE[INV_OUT] = 1
	CRCPE[CRC_INIT] = 0xFFFF

To enable the CRC check for a given BD, set the [CRC_EN] bit of the BD. Once set, the MAPLE-B executes the CRC check on the Hard Output results, and only then outputs the data to system memory. The MAPLE-B gives a CRC fail indication in the [ERROR_STATUS] field of the TVPE BD. If CRC check passes no indication is given.

If CRC stopping criteria is enabled (see **Section 26.3.2.1.1.5**, *Stopping Criteria*), the MAPLE-B executes a CRC check after every Turbo full iteration, regardless of the [CRC_EN] bit of the TVPE BD.

Note: The CRC checks assumes ascending byte/bit ordering of the hard outputs (see **Section 26.3.2.1.3**, *TVPE Output Data*, on page 26-73). Any other byte/bit ordering results in a CRC fail.

Note: Although for 3GLTE operation mode and ‘Sub-Block Interleaved’ input data structure, the MAPLE-B supports all block sizes ranging from 1 to 40, the CRC check are executed on the QPP size (see **Section 26.3.1.4.11.2**, *3GLTE Supported Block Sizes (BS)*, on page 26-53), which is the size used for the CRC calculations during the encoding.

Note: For 3GPP operation mode CRC checks can be executed only for Block Sizes aligned to 8. Any other Block Size causes the CRC check to fail.

26.3.2.1.1.5 Stopping Criteria

The MAPLE-B offers 2 types of stopping criteria which allow it to stop decoding before reaching the [MAX_ITER] values specified in the TVPE BD. If stopping criteria is not enabled the MAPLE-B executes the number of iterations as specified in the [MAX_ITER] field. The [MIN_ITER] field describes the minimum number of iterations executed by the MAPLE-B when any of the stop criteria is enabled.

26.3.2.1.1.5.1 Aposteriori Output Quality Based Stopping Criteria

In order for the MAPLE-B to work with the Aposteriori Output quality criteria, the following parameters/TVPE registers must be configured:

- MTSCCP[S_APP] - Aposteriori Stop Criteria enable bit. This bit must be set.
- TVPE register: TVAQCR[AQTH] - Threshold value, to be compared to the Aposteriori results in each step during the iteration. If The Aposteriori calculations values are greater that the AQTH value, an internal counter for the current iteration is incriminated.
- MTSCCP[STOP_CNT] - Quality counter, a fraction moving between 1/16 to 16/16 which determines the relative number of aposteriori decisions, out of the total number of trellis-steps, that should pass the threshold before the decoding operation is stopped. For WiMAX decoding processing operation mode (Duo binary decoding), the total number of Trellis-steps equals 1/2 the number of decoded bits.

For each Trellis step, the TVPE calculates the following to determine if Aposteriori Output quality has been reached:

For Binary decoding (3Gxx):

if ((| L_n[P(0)] - L_n[P(1)] | > TVAQCR[AQTH]) **&** (| L_{n+1}[P(0)] - L_{n+1}[P(1)] | > TVAQCR[AQTH]))

counter = counter + 2

else if ((| L_n[P(0)] - L_n[P(1)] | > TVAQCR[AQTH]) | (| L_{n+1}[P(0)] - L_{n+1}[P(1)] | > TVAQCR[AQTH]))

counter = counter + 1;

For Duo Binary decoding (WiMAX):

1. find max APP (APP_m) by MAX(|APP_00|, |APP_01|, |APP_10|, |APP_11|)
2. if (|APP_m - APP_cm| > TVAQCR[AQTH])

counter = counter + 2

where:

APP_cm =

if (*APP_m* = *APP_00*) then APP_11
if (*APP_m* = *APP_01*) then APP_10
if (*APP_m* = *APP_10*) then APP_01
if (*APP_m* = *APP_11*) then APP_00

Note: For the 3Gxx operation modes that use the Zero Tail trellis termination method, the number of trellis steps is always BS + 3.

The stopping criteria are only checked starting from the iteration number defined in the [MIN_ITER] field, defined in the TVPE BD, that is, the [MIN_ITER] field dictates the minimum number of iterations executed by the TVPE regardless of the Aposteriori Output Quality Based Stopping criteria.

The MAPLE-B performs the AQ checks after every TVPE internal indication signaling that a new iteration is completed. For small TVPE Block Sizes, where the execution time of one iteration is short, the MAPLE-B does not have enough time to perform the check. Therefore, **Table 26-36** describes the minimum Block Sizes which the MAPLE-B can support AQ based stop criteria. If the Block Size is smaller than the minimum described in **Table 26-36**, the MAPLE-B performs [MAX_ITER] number of iterations although AQ based stopping criteria is enabled.

Table 26-36. Minimum Block Size supporting AQ Based Stopping Criteria

Block Size (BS)	Executed by MAPLE-B if AQ stop criteria is enabled
BS > 400	MAPLE-B performs AQ based stop criteria
BS ≤ 400	MAPLE-B performs the number of iterations as described in MAX_ITER field of the TVPE BD.

The [CMPLT_RSN] status field in the TVPE BD, indicates whether the TVPE completed its decoding process due to stopping criteria, or due to reaching the maximum iteration number given in the [MAX_ITER] field of the TVPE BD.

The [ITER_CNT] status field in the TVPE BD indicates the actual number of iterations executed for the current job. If no stopping criteria is enabled or if the stopping criteria has not reach its threshold, the value of the [ITER_CNT] is equal to the value of the [MAX_ITER]. If the stopping criteria reaches its threshold before the number of iterations specified in the [MAX_ITER] has executed, the [ITER_CNT] field contains the actual number of iteration executed until reaching the stopping criteria threshold.

Note: Although for 3GLTE operation mode and ‘Sub-Block Interleaved’ input data structure, the MAPLE-B supports all possible block sizes ranging from 1 to 6144, the AQ Based Stopping criteria assumes the Block Size is the QPP size (see **Section 26.3.1.4.11.2**, *3GLTE Supported Block Sizes (BS)*, on page 26-53), and therefore relates its stop criteria calculations to the QPP size.

Simulation results shown in **Table 26-37** suggest the recommended range of values to use for the [AQTH] field.

Table 26-37. Recommended range of values for the [AQTH]

Operation Mode	Recommended value range for [AQTH][1] field
3GLTE	100–150
WiMAX	200–300

1. The higher the number is, the better the decoding quality is and the required number of iterations.

26.3.2.1.1.5.2 CRC Check Based Stopping Criteria

The CRC check based stopping criteria means that the MAPLE-B runs CRC check on the Hard Output results of the TVPE after each full iteration. If the CRC check has passed, the MAPLE-B stops the TVPE operation and outputs the Hard Output results to the system memory.

If the CRC check based stopping criteria is enabled the CRC checks on the Hard Outputs are executed when the minimum number of iterations is reached. If minimum number of iterations is set to zero ([MIN_ITER] = 0), then CRC checks on Hard Outputs start from first iteration.

If the CRC check based stopping criteria is enabled and if the [MAX_ITER] field of the TVPE BD, which sets the maximum number of iterations to be executed by the TVPE, is reached without passing the CRC check, the MAPLE-B stops the decoding iterations and gives a CRC fail indication in the status field of the TVPE BD ([ERROR_STATUS]).

The MAPLE-B performs the CRC checks after every TVPE internal indication signaling that a new iteration is completed. For small TVPE Block Sizes, where the execution time of one iteration is short, the MAPLE-B does not have enough time to perform the CRC check on the hard outputs of the current iteration before the hard outputs of the next iteration overwrites them. Therefore, **Table 26-38** describes the minimum Block Sizes which the MAPLE-B can support CRC stop criteria. If the Block Size is smaller than the minimum described in **Table 26-38**, the MAPLE-B performs [MAX_ITER] number of iterations although CRC stopping criteria is enabled.

Table 26-38. Minimum Block Size Supporting CRC Stop Criteria

Block Size (BS)	Executed by MAPLE-B if CRC Stop Criteria Is Enabled
BS ≥ 400	MAPLE-B performs CRC stop criteria
BS < 400	MAPLE-B performs the number of iterations as described in MAX_ITER field of the TVPE BD.

Enabling the CRC check based stopping criteria is done by setting the [S_CRC] bit of the MTVCP parameter in the parameter RAM.

The CRC polynomial configuration for the CRC based stop criteria is determined by the [CRC_POLY] field of the TVPE BD. Setting a CRC polynomial also sets a number of CRCPE parameters as described in **Section 26.3.2.1.1.4**, *CRC Check*.

Note: Although for 3GLTE operation mode and Sub-Block Interleaved input data structure, the MAPLE-B supports all block sizes ranging from 1 to 40, the CRC check are executed on the QPP size (see **Section 26.3.1.4.11.2**, *3GLTE Supported Block Sizes (BS)*, on page 26-53), which is the size used for the CRC calculations during the encoding.

Note: For 3GPP operation mode CRC checks can be executed only for Block Sizes aligned to 8. Any other Block Size will cause the CRC check to fail.

Note: For 3GPP2 operation mode CRC checks are not allowed as all defined Block Sizes are not aligned to 8 bits.

26.3.2.1.1.5.3 RISC Engine Configuration for Stopping Criteria Enablement

The MAPLE-B includes two internal RISC engines that are responsible for various tasks such as parsing the BDs, controlling the MAPLE-B internal DMA engines, activating the PEs, and so forth. Enabling any of the above stopping criteria requires special treatment from the RISC engines, which are required to monitor the TVPE execution and to stop its activity once the given stop condition is fulfilled. To allow that, the mode parameter must be initialized during the MAPLE-B activation as described in the API User's Manual. Configuration of the mode parameter changes the default task partitioning between the RISC engines, thus allowing the required controllability over the TVPE stop criteria execution.

Avoiding the mode configuration while enabling either of the stop criteria may result in incorrect behavior of the stop criteria.

26.3.2.1.1.5.4 Hard and Soft output results when Stopping Criteria is enabled

In order to minimize the TVPE throughput degradation during the Stopping Criteria operation, the RISC engine perform the Stopping Criteria check (CRC or AQ) of iteration X while the TVPE is working on iteration X+1 ([MIN_ITER] $\leq$ X < [MAX_ITER]). Therefore when Stopping Criteria check has passed and the RISC engine orders the TVPE to stop, the TVPE will stop after the completion of the current iteration, which is iteration X+1. Since the TVPE maintains double output buffer for the Hard output results, on completion of iteration X+1, the TVPE holds the Hard output results of both, iteration X and iteration X+1, and therefore, if enabled by the [HOE] field of the TVPE BD, the MAPLE-B will output the Hard results of iteration X, which passed the Stopping criteria check. For the Soft results, the TVPE does not maintain double buffer, therefore, if enabled by the [SOE] field of the TVPE BD, the MAPLE-B will output the Soft results of iteration X+1 although it was iteration X which passed the Stopping criteria check. Detecting cases where the TVPE stopped due to Stopping criteria is done using the [CMP_RSN] field of the TVPE BD.

Note: The [ITER_CNT] status field in the TVPE BD always indicates the actual number of Turbo iterations executed by the TVPE. If Stopping criteria is enabled, and the TVPE stops due to Stopping Criteria, the [ITER_CNT] field indicated the number of iteration X + 1, although the Hard results are taken from iteration X.

26.3.2.1.2 Viterbi Decoding Operation

The MAPLE-B is capable of Viterbi decoding ([ALG]=1) with the following configurable parameters:

- Constraint Length of 5,6,7,8 and 9 ([VIT_K]= 0,1,2,3,4, respectively).
- Encoding rate of 1/2, 1/3, and 1/4 ([ENC_RATE]=0,1,2, respectively).
- Tail Biting/Zero Tail trellis termination ([ZTTB] = 0,1, respectively)
- Supported block sizes of up to:
  - 8192 for K= 5
  - 2688 for K = 7
  - 672 for K = 9

Note: The maximum block sizes include the tail bits in case of zero tail decoding. It means that the actual maximum decoded block sizes in case of zero tail decoding (ZTTB=1) are 8188, 2582 and 664 for K = 5, K = 7 and K = 9 respectively.

26.3.2.1.2.1 Viterbi Processing—General

The Viterbi processing in the MAPLE-B is not standard related, and is fully configurable (including polynomials) regardless of the MAPLE-B operation mode. It is up to the host to configure all Viterbi related parameters/registers to process Viterbi decoding.

Viterbi processing executes internally using three steps:

- Feed forward calculations
- Find maximum likelihood path
- Trace back calculation.

Each stages is explained in the following sections.

26.3.2.1.2.1.1 Viterbi Feed Forward

During this stage, the TVPE calculates 2^{k-1} path Metrics, each with length n, where K is the Constraint Length and n is the block size.

Figure 26-13 illustrates an example of a path metric (PM) calculation where path metric $PM_{n-1,s}$ and branch metric (BM) $BM_{n-1,s}$ are used for calculating the $PM_{n,s}$.

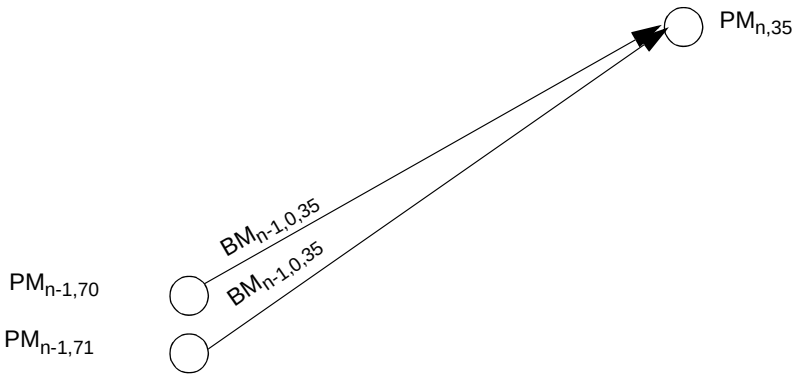


Figure 26-13. $PM_{n,35}$ Calculation

where:

PM $_{n,s,}$ - Path Metric of Viterbi Step n within the trellis. The s represent the current state. There are 2^{k-1} states in each step, where K is the Constraint Length.

BM $_{n-1,x,s}$ - Branch Metric for the transition calculation from step n-1 to step n. The X is either 0 or 1, and it represents the input bit to the Viterbi encoder.

To calculate the PM of trellis step n, state 35 (**Figure 26-13**), the PM of trellis step n-1, states 70 and 71 are needed.

The BMs are the correlations between the received parity bytes and the expected parity bytes.

$$BMn,0,d = Parity_{0,n} * poly0_{<(d*2+0)\bmod \text{num of pms}>} + Parity_{1,n} * poly1_{<(d*2+0)\bmod \text{num of pms}>} + Parity_{2,n} * poly2_{<(d*2+0)\bmod \text{num of pms}>} + Parity_{3,n} * poly3_{<(d*2+0)\bmod \text{nuim of pms}>}$$

$$BMn,1,d = Parity_{0,n} * poly0_{<(d*2+1)\bmod \text{num of pms}>} + Parity_{1,n} * poly1_{<(d*2+1)\bmod \text{num of pms}>} + Parity_{2,n} * poly2_{<(d*2+1)\bmod \text{num of pms}>} + Parity_{3,n} * poly3_{<(d*2+1)\bmod \text{num of pms}>}$$

where d is destination state and pms is path metric states.

$Parity_{3,n} * poly3_{<(d*2+1)\bmod \text{num of pms}>}$The expected parity bytes are the parities that are calculated on the source state using the parity polynomials described in **Figure 26-14**.

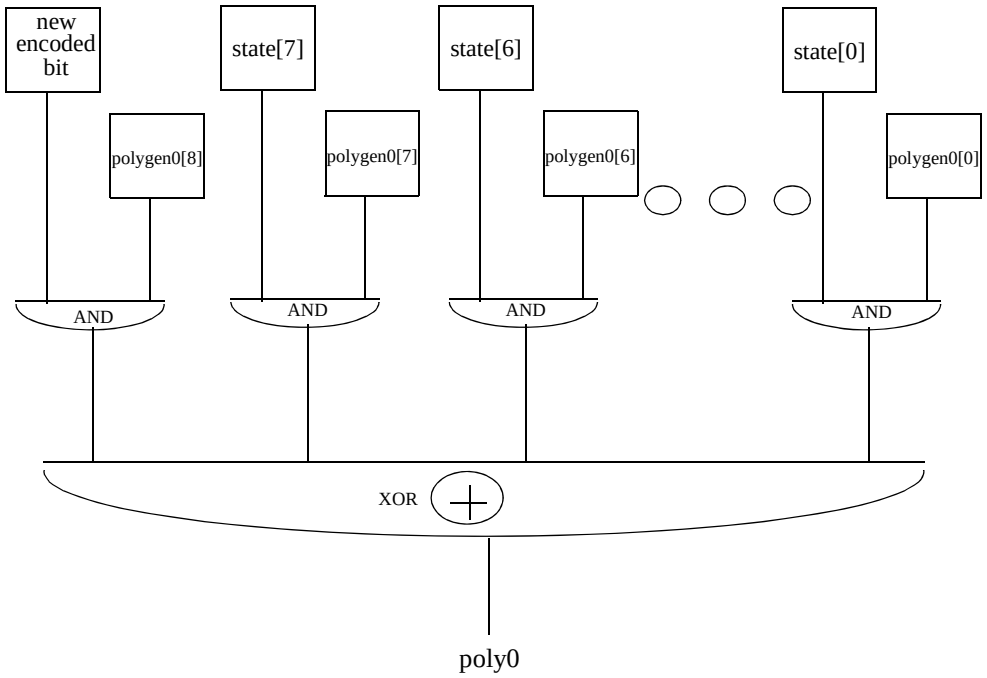


Figure 26-14. Calculation of a parity of a step

The Polynomial Generation fields **TVVPVGxCR[POLYGENx]** are used for calculating the parity bits per state.

poly'z'$_{<x>}$ =
(new_encoded_bit & **POLYGEN'z'[8]**) ^
(x[7] & **POLYGEN'z'[7]**) ^
(x[6] & **POLYGEN'z'[6]**) ^
(x[5] & **POLYGEN'z'[5]**) ^
(x[4] & **POLYGEN'z'[4]**) ^
(x[3] & **POLYGEN'z'[3]**) ^
(x[2] & **POLYGEN'z'[2]**) ^
(x[1] & **POLYGEN'z'[1]**) ^
(x[0] & **POLYGEN'z'[0]**)
where 'z' = 0...3.

Note: If poly3$_{<x>}$ is not used (Rate > 1/4) and poly2$_{<x>}$ is not used (Rate > 1/3), they should be set to 0x0000, otherwise, the Viterbi Feed-Forward will not work properly.

Note: The value of the POLYGEN fields must not exceed the $2^k - 1$ value, where K is the Viterbi Constraint Length.

The TVPE maintain single set of polynomials configuration. In order to support dynamic switching of multiple Viterbi polynomial configurations, the MAPLE-B maintains three sets of polynomial configurations parameters which can be dynamically (per TVPE BD) be replaced by

the MAPLE-B firmware. The control of which set is being used for each TVPE BD is done by the [VIT_SET] filed of the TVPE BD. Setting the [VIT_SET] field to ‘00’ causes the MAPLE-B to use the internal TVPE registers configured by the user (TVVPVG1CR). This configuration should be used if no dynamic switching of Viterbi polynomial configuration is required. Setting the [VIT_SET] to ‘01’, ‘10’ or ‘11’ causes the MAPLE-B to use its MTVPVS1CxP, MTVPVS2CxP or MTVPVS3CxP parameters sets respectively.

The Path Metric is calculated using compare-select between the addition of the source states and their respective branch matrices as follows:

$$PM_{n,d} = \text{compare\_2\_select}\left((PM_{n-1,(d*2+0)\bmod pms} + BM_{n,0,d}), (PM_{(n-1,(d*2+1)\bmod pms} + BM_{n,1,d})\right); \text{for } n>0$$

Where:

- n is the step index,
- pms is 2^(VIT_K-1).
- d identifies the destination states 0<=d<=255.
- $Parity_{p,n}$ = base(PFp) + n
- compare_2_select(u,v) = (msbit of U) ^ (msbit of V) ^ ((lsbits of u) > (lsbits of v))
- $PM_{0,d}$ is the $PM_{<d>}$ field in the $TVVPM_{<d>}R$ register before execution.

26.3.2.1.2.1.2 Maximum Calculations

The TVPE compares the final Path Metrics to find the state with the maximum path metric value. The PM value fields compared depend on the [VIT_K] field and are defined in **Table 26-39**.

Table 26-39. PM Values to be Compared for Finding the Maximum

VIT_K	K	PMs To Be Compared
001	5	PM_0 -> PM_15
010	6	PM_0 -> PM_31
011	7	PM_0 -> PM_63
100	8	PM_0 -> PM_127
101	9	PM_0 -> PM_255

26.3.2.1.2.1.3 Trace-Back Calculations

The TVPE, starting from a given trellis step traces back the winning path according to an internal history buffer. While tracing back the winning path, the TVPE generates the Hard Output bits, which are then transferred to external memory.

26.3.2.1.2.2 Zero Tail Viterbi Processing

For Zero Tail Viterbi processing ([ALG]=1, [ZTTB]=1), the MAPLE-B performs the following sequence of operations:

1. Initialize state zero of the trellis step with maximum PM value.
2. Initiate Feed Forward calculations starting from the first step until the last step of the trellis.
3. Find the state with the maximum PM value.
4. According to the Maximum calculation result and the value of the [TBZE] bit in the TVPE BD, the MAPLE-B executes the following:

   If ([TBZE]=0 and Max_state = 0) //Max_state is the result state from the maximum calculations

   Start Trace-Back calculation from state zero

   Else if ([TBZE]=0 and Max_state!= 0)

   Start Trace-Back calculation from state zero and set ZT_MAX // ZT_MAX is a status bit in the TVPE BD

   Else if ([TBZE]=1 and Max_state = 0)

   Start Trace-Back calculation from state zero

   Else if ([TBZE]=1 and Max_state!= 0)

   Start Trace-Back calculation from Max_state and set [ZT_MAX]
5. If the [HOE] bit of the TVPE BD is set, then Hard Outputs bits are generated during the trace back calculations and the MAPLE-B outputs them to system memory according to the pointers given in the BD.

26.3.2.1.2.3 Tail Biting Viterbi Processing (WAVA*)

For Tail Biting Viterbi processing ([ALG]=1, [ZTTB]=0), the MAPLE-B perform the WAVA* (Wrap-Around-Viterbi-Algorithm). The sequence of events executed by the MAPLE-B when implementing Viterbi Tail biting using WAVA* is:

1. Initialize all path metric (PM) values to zero.
2. Initiate Feed Forward calculation on the trellis
3. Initiate additional Feed Forward calculations on the same data, but without initializing the PM values. The additional Feed Forward calculation, this time with initial PM values which are the results of the first Feed Forward calculation will improve PM results.
4. Initiate additional Feed Forward calculations on the same data, but without initializing the PM values. The additional Feed Forward calculation, this time with initial PM values which are the results of the first two Feed Forward calculation will improve PM results.
5. Find maximum PM value.

6. Find the initial state of that trellis (without generating Hard Outputs).
7. Use the last state found in the previous step as the starting point for the actual Trace-Back that generates the Hard Outputs.

Figure 26-15 describes the WAVA* high level flow:

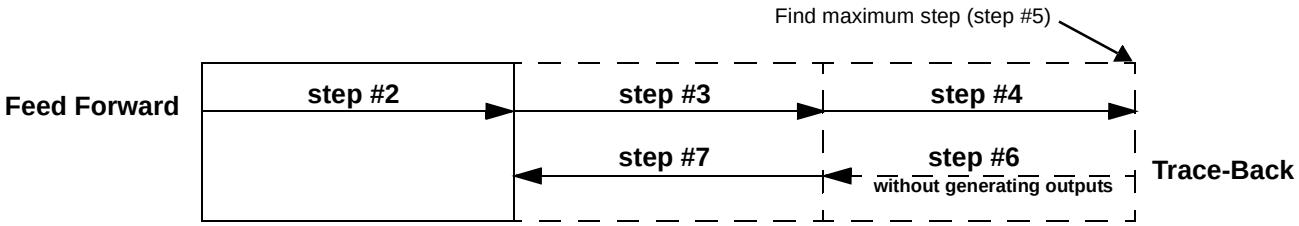


Figure 26-15. Viterbi Tail-Biting WAVA* flow

26.3.2.1.2.4 Viterbi Large Blocks Partitioning Support

The MAPLE-B allows Viterbi decoding of blocks with size of up to 32KB using multiple BDs scheme and with the following limitation:

- Zero tail decoding only
- 'Direct' input data structure only
- Hard Outputs byte/bit ordering (see Section 26.3.2.1.4.3) supported for Viterbi large block sizes is either byte/bit ascending order or byte/bit descending order. The Hard Outputs data structure can not be with ascending byte order and descending bit order or vice versa.

Such multiply BDs scheme must be implemented for any block which is bigger than the sizes described in the **Table 26-40**:

Table 26-40. Viterbi maximum Block sizes Executed in Single Session with No Partitioning

Viterbi K	Max Block Size for Single Session[1]
5	8192
6	5376
7	2688
8	1344
9	672

1. The size includes the zero tail bits.

When Viterbi decoding is required for block which are larger than described in **Table 26-40**, the host must split the block into several chunks and each chunk should be described in different BD. When partitioning the large block into several chunks, the size of each chunk must not exceed the sizes as described in **Table 26-41**:

Table 26-41. Viterbi Maximum Chunk Sizes During Large Block Partitioning Scheme

Viterbi K	Maximum Chunk for Partitioning
5	8192
6	5376

Table 26-41. Viterbi Maximum Chunk Sizes During Large Block Partitioning Scheme

Viterbi K	Maximum Chunk for Partitioning
7	2688
8	1280
9	640

To maximize decoding quality, the host must partition the large block into overlapping chunks, thus enabling “warm-up” sections for both the Feed-Forward stage and the Trace-Back stage. The host can configure the size of the “warm-up” section in each block using the BUF_SIZE field of the TVPE BD. It must be a multiple of 64. **Figure 26-16** describes an example of 24 Kbit (24576) viterbi code and its partitioning into several chunks in the case of K=5.

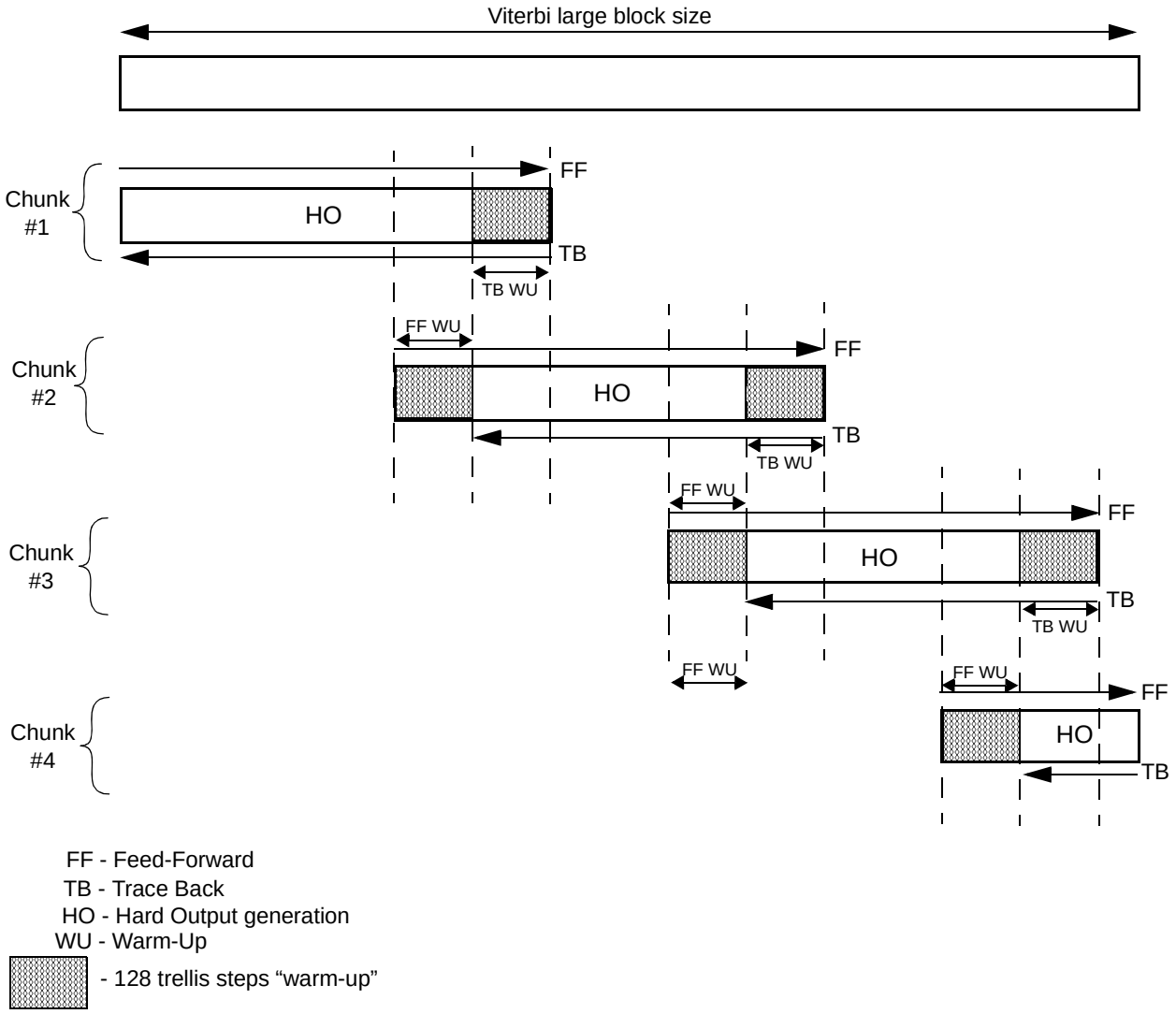


Figure 26-16. Viterbi Large Blocks Size Partitioning Flow

The example assumes the following:

- Encoding rate: 1/2
- ‘Direct’ input data structure
- Hard outputs are required at address 0xc0050000
- First input vector (even index number, interlaced P0 and P1) is placed at address 0xc0000000
- Second input vector (odd index number, interlaced P0 and P1) is placed at address 0xc0020000

Table 26-42 describes the relevant fields in TVPE BD for each chunk as described in **Figure 26-16**:

Table 26-42. TVPE BD Fields of Viterbi Large Block Size Partitioned into Chunks

Chunk #	Field name	Value	Comments
1	VIT_K	001	K = 5
	ZTTB	0	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	ADJ	0	The input vectors are non adjacent
	ALG	1	Viterbi decoding
	IN_D_STRCT	0	‘Direct’ input data structure
	BS	0x2000	Block size of 8192
	HOE	1	Enable hard outputs
	BUF_SIZE	1	For the first chunk only, the value of this field must be set to ‘1’ in order to allow full trace-back
	HRD_RSLT_ADDR	0xc0050000	
	BASE_ADDR	0xc0000000	
	ADDR_OFFSET_0	0x0000	First vector is BASE_ADDR+0x00000
	ADDR_OFFSET_1	0x0800	Second vector is BASE_ADDR+0x20000

Table 26-42. TVPE BD Fields of Viterbi Large Block Size Partitioned into Chunks (Continued)

Chunk #	Field name	Value	Comments
2	VIT_K	001	K = 5
	ZTTB	0	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	ADJ	0	The input vectors are non adjacent
	ALG	1	Viterbi decoding
	IN_D_STRCT	0	'Direct' input data structure
	BS	0x2000	Block size of 8192
	HOE	1	Enable hard outputs
	BUF_SIZE	0x100	Equals 256. Indicates the Feed-Forward warm-up and Trace back warm-up required by MAPLE-B
	HRD_RSLT_ADDR	0xc00503E0	The start of the hard outputs for the second chunk should be (8192 - 256)/8 = 7936/8 = 0x3E0
	BASE_ADDR	0xc0000000	
	ADDR_OFFSET_0	0x00F0	The second chuck of input should start at offset of 8192 - 2*256 = 7680. Each input is represented by 4 bytes. Each vector include either Odd or Even index. Hence (7680*4/2)/64=120 =0xF0
	ADDR_OFFSET_1	0x08F0	The second chuck of input should start at offset of 8192 - 2*256 = 7680. Each input is represented by 4 bytes. Each vector include either Odd or Even index. Hence (7680*4/2)/64=120 =0xF0
3	VIT_K	001	K = 5
	ZTTB	0	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	ADJ	0	The input vectors are non adjacent
	ALG	1	Viterbi decoding
	IN_D_STRCT	0	'Direct' input data structure
	BS	0x2000	Block size of 8192
	HOE	1	Enable hard outputs
	BUF_SIZE	0x100	Equals 256. Indicates the Feed-Forward warm-up and Trace back warm-up to be executed by MAPLE-B
	HRD_RSLT_ADDR	0xc00507A0	The start of the hard outputs for the third chunk should be (8192*2 - 256*3)/8 = 15616/8 = 0x7A0
	BASE_ADDR	0xc0000000	
	ADDR_OFFSET_0	0x01E0	The third chuck of input should start at offset of 8192*2 - 256*4= 15360. Each input is represented by 4 bytes. Each vector include either Odd or Even index. Hence (15360*4/2)/64=480 =0x1E0
	ADDR_OFFSET_1	0x09E0	The third chuck of input should start at offset of 8192*2 - 256*4 = 15360. Each input is represented by 4 bytes. Each vector include either Odd or Even index. Hence (15360*4/2)/64=480 =0x1E0

Table 26-42. TVPE BD Fields of Viterbi Large Block Size Partitioned into Chunks (Continued)

Chunk #	Field name	Value	Comments
4	VIT_K	001	K = 5
	ZTTB	1	Zero Tail decoding
	ENC_RATE	00	Encoding rate: 1/2
	ADJ	0	The input vectors are non adjacent
	ALG	1	Viterbi decoding
	IN_D_STRCT	0	'Direct' input data structure
	BS	0x0600	The last chunk size equals: 24*1024 - (3*8192 - 6*256) = 1536 = 0x600
	HOE	1	Enable hard outputs
	BUF_SIZE	0x100	Equals 256. Indicates the Feed-Forward warm-up and Trace back warm-up to be executed by MAPLE-B
	HRD_RSLT_ADDR	0xc0050B60	The start of the hard outputs for the forth chunk should be (8192*3 - 256*5)/8 = 23296/8 = 0xB60
	BASE_ADDR	0xc0000000	
	ADDR_OFFSET_0	0x02D0	The forth chuck of input should start at offset of 8192*3 - 256*6= 23040. Each input is represented by 4 bytes. Each vector include either Odd or Even index. Hence (23040*4/2)/64=720=0x2D0
	ADDR_OFFSET_1	0x0AD0	The forth chuck of input should start at offset of 8192*3 - 256*6 = 23040. Each input is represented by 4 bytes. Each vector include either Odd or Even index. Hence (23040*4/2)/64=720=0x2D0

26.3.2.1.3 TVPE Output Data

The TVPE can generate two types of output data ordered internally according to two configuration bits. The following sections describe the output data types and output data structure.

26.3.2.1.3.1 Output Data Types

The TVPE is capable of generating the following types of data:

- Soft Output data
- Hard Output bits

26.3.2.1.3.2 Soft/Extrinsic Output Data

Soft/Extrinsic Output data generation is relevant only for Turbo decoding, and is enabled by setting the [SOE] bit in the TVPE BD. This option is relevant if additional processing (external to the MAPLE-B) is needed on the data before calculating the final Hard Outputs, or for debug purposes.

The TVPE generates one of two types of soft outputs:

- If [SOE] = 01, the TVPE generates Soft Aposteriori output.
- If [SOE] = 10, the TVPE generates Extrinsic output.

26.3.2.1.3.2.1 Soft Output Data

The TVPE generates the Soft Outputs differently when executing Duo-Binary Turbo decoding (WiMAX standard operation mode) or Binary Turbo decoding (3GLTE, 3GPP, 3GPP2). **Table 26-43** describes the Soft Output data for both cases.

Table 26-43. Soft Output Description

Turbo Code	Structure	Notes
Binary	APP[0],APP[1],...,APP[BS[1]-1]	Each APP is the result of subtracting the Aposteriori probability of receiving 1 from the Aposteriori probability of receiving 0: APP[x] = Ln[$P_x(1)/P_x(0)$];
Duo-Binary	MLLR_1[0],MLLR_2[0],MLLR_3[0],MLLR_1[1],MLLR_2[1], MLLR_3[1]... MLLR_1[BS/2-1],MLLR_2[BS/2-1],MLLR_3[BS/2-1],	Each stage of duo-binary decoding results in 4 aposteriori values (APP_00, APP_01, APP_10, APP_11). The MLLR_x values are derived from the APP_xx values as follows: $MLLR1_{xy}$ = Ln[APP_00/APP_01] $MLLR2_{xy}$ = Ln[APP_00/APP_10] $MLLR3_{xy}$ = Ln[APP_00/APP_11]

1. BS: block size.

The following equations describe how to reconstruct the approximated APPxx values from the MLLR_x values for WiMAX decoding:

$$APP\_00[x] \sim= max(MLLR\_1[x], MLLR\_2[x], MLLR\_3[x], 0)$$
$$APP\_01[x] \sim= max(-MLLR\_1[x], -MLLR\_1[x] + MLLR\_2[x], -MLLR\_1[x] + MLLR\_3[x], 0)$$
$$APP\_10[x] \sim= max(-MLLR\_2[x], -MLLR\_2[x] + MLLR\_1[x], -MLLR\_2[x] + MLLR\_3[x], 0)$$
$$APP\_11[x] \sim= max(-MLLR\_3[x], -MLLR\_3[x] + MLLR\_1[x], -MLLR\_3[x] + MLLR\_2[x], 0)$$

The size of a single element of the Soft Output data structure for both, Binary and Duo binary decoding (APP[x] and MLLR_x[y], respectively), is 16 bits.

Upon job completion, the MAPLE-B outputs the Soft Output data to system memory in the location specified in the [SFT_RSLT_ADDR] field of the TVPE BD. The structure of the Soft Output results differs according to the decoding method (Binary for 3Gxx standard operation modes and Duo binary for WiMAX standard operation mode). **Figure 26-17** gives an example of a Soft Output results buffer for the 3Gxx operation modes in a 128 bit wide system memory:

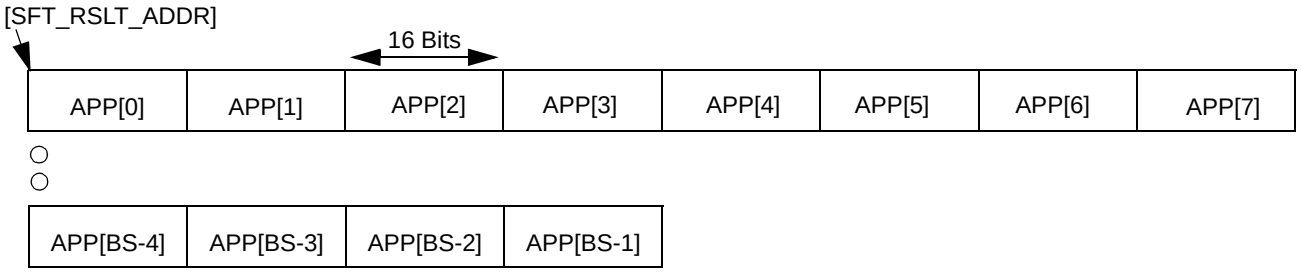


Figure 26-17. Example of Soft Output Results Buffer for 3Gxx Operation Modes in a 128 Bit System Memory

Figure 26-18 gives an example of a Soft Output results buffer for the WiMAX operation mode in a 128 bit wide system memory:

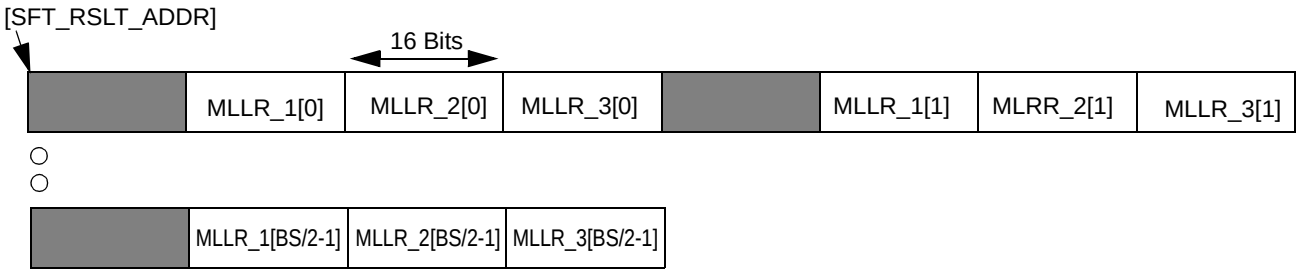


Figure 26-18. Example of Soft Output Results Buffer for WiMAX Operation Mode in a 128 Bit System Memory

26.3.2.1.3.2.2 Extrinsic Output Data

The TVPE is capable of generating the Extrinsic data information instead of the Aposteriori data. The Extrinsic data is different when executing Duo-Binary Turbo decoding (WiMAX standard operation mode) and Binary Turbo decoding (3GLTE, 3GPP, 3GPP2). **Table 26-44** describes the Extrinsic Output data for both cases.

Table 26-44. Extrinsic Output Description

Turbo Code	Structure	Notes
Binary	EXT[0],EXT[1],...,EXT[BS[1]-1]	Each EXT is the result of subtracting from the APP value the EXT value from the previous iteration and the Systematic bit (its soft representation: $EXT_i[x] = APP_i[x] - EXT_{i-1}[x] - SD[x]$. (i is the current iteration)
Duo-Binary	EXT_1[0],EXT_2[0],EXT_3[0],EXT_1[1],EXT_2[1],EXT_3[1]... EXT_1[BS/2-1],EXT_2[BS/2-1],EXT_3[BS/2-1],	Each set of 3 Extrinsic values hold information on pair of bits. The equations for each of the EXT values for iteration #i is: $EXT\_1_{xy,i} = MLLR\_1_{xy} - EXT\_1_{xy,i-1} - 2*SDb$ $EXT\_2_{xy,i} = MLLR\_2_{xy} - EXT\_2_{xy,i-1} - 2*SDa$ $EXT\_3_{xy,i} = MLLR\_3_{xy} - EXT\_3_{xy,i-1} - 2*SDa - 2*SDb$

1. BS: block size.

The size of a single element of the Extrinsic Output data structure for both, Binary and Duo binary decoding (EXT[x] and EXT_x[y], respectively), is 16 bits.

Upon job completion, the MAPLE-B outputs the Extrinsic Output data to system memory in the location specified in the [SFT_RSLT_ADDR] field of the TVPE BD. The structure of the extrinsic Output results differs according to the decoding method (Binary for 3Gxx standard operation modes and Duo binary for WiMAX standard operation mode). **Figure 26-19** gives an example of an extrinsic Output results buffer for the 3Gxx operation modes in a 128 bit wide system memory:

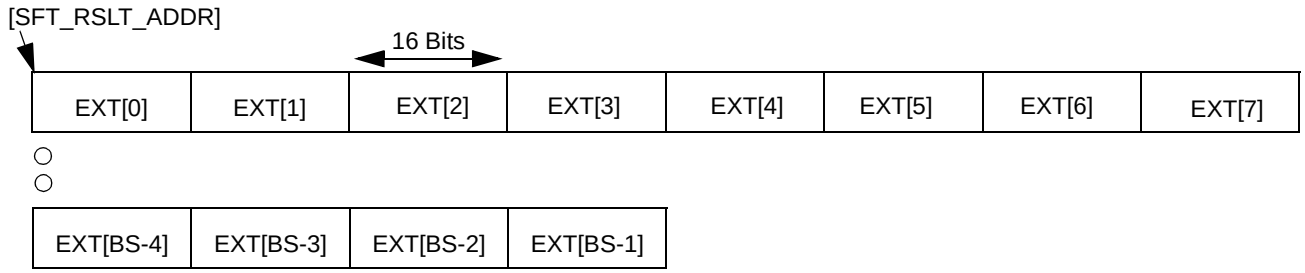


Figure 26-19. Example of Extrinsic Output Results Buffer for 3Gxx Operation Modes in a 128 Bit System Memory

Figure 26-20 gives an example of a Extrinsic Output results buffer for the WiMAX operation mode in a 128-bit wide system memory:

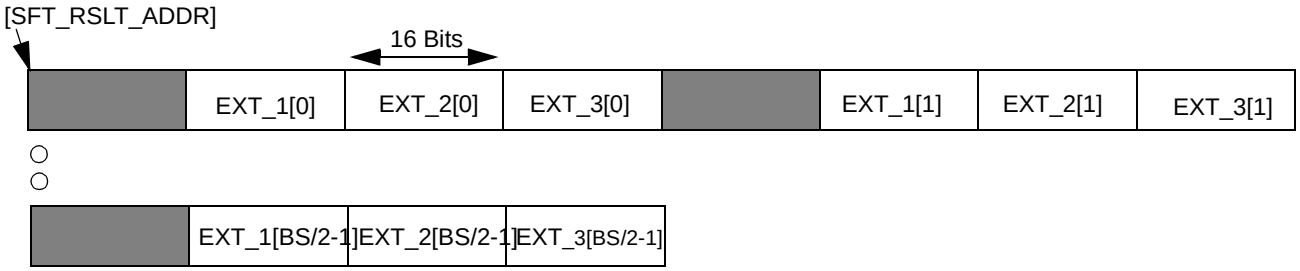


Figure 26-20. Example of Extrinsic Output Results Buffer for WiMAX Operation Mode in a 128 Bit System Memory

Note: Enabling the MAPLE-B to output its soft Turbo result consumes greater DMA/riscs resources than enabling it to output the hard result (**Section 26.3.2.1.3.3**, *Hard Output Data*, on page 26-76"). For optimal throughput performance it is recommended to output the hard result only.

26.3.2.1.3.3 Hard Output Data

The Hard Output data generation is relevant for both Turbo and Viterbi decoding and is enabled by setting the [HOE] bit in the TVPE BD.

The Hard Outputs for the Turbo decoding are generated from the Soft Outputs as follows:

- For Binary decoding (all standards except WiMAX):
  - if APP[i] > 0 then the Hard Output bit with the index [i] equals 1, otherwise it equals 0.
- For Duo-Binary code (WiMAX):
  - reconstruct APP_xx[i] from the MLLR_x.
  - find: Max (APP_00[i], APP_01[i], APP_10[i], APP_11[i]).
  - The APP_ab[i] winner means bits [2i] and [2i + 1] equal a and b respectively.

Upon job completion, the MAPLE-B outputs the Hard Output data to system memory in the location specified in the [HRD_RSLT_ADDR] field of the TVPE BD.

26.3.2.1.4 Output Data Structure

The TVPE supports multiple output data structures for the Hard Output bits (if enabled):

26.3.2.1.4.1 Byte Ordering

The Hard Outputs byte order is programmable. If the TVPEC0R[DOSBY] bit is set, then the Hard Outputs are arranged in bytes in ascending order as illustrated in **Figure 26-21**.

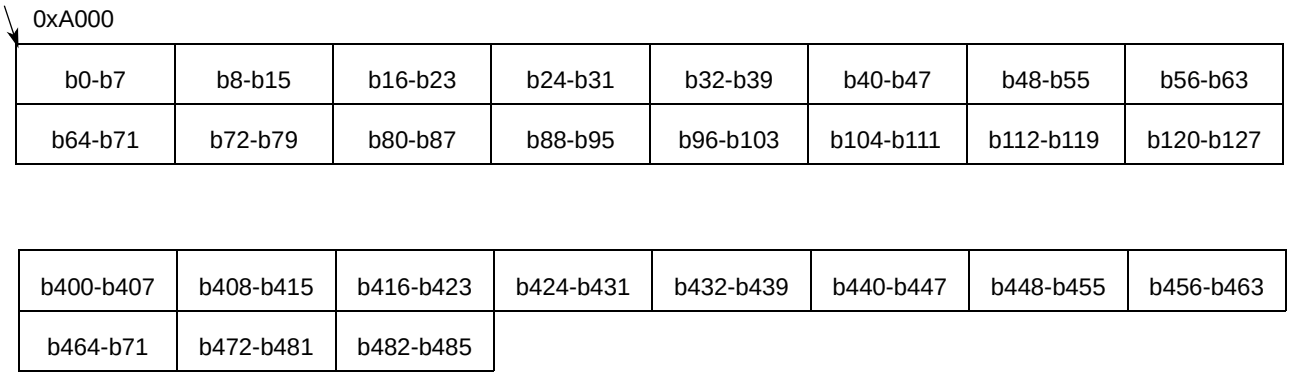


Figure 26-21. Ascending Byte Ordering (DOSBY=1) of a 486 Hard Bits Output Block

If the TVPEC0R[DOSBY] bit is cleared, then the Hard Outputs are arranged in bytes in descending order as illustrated in **Figure 26-22**.

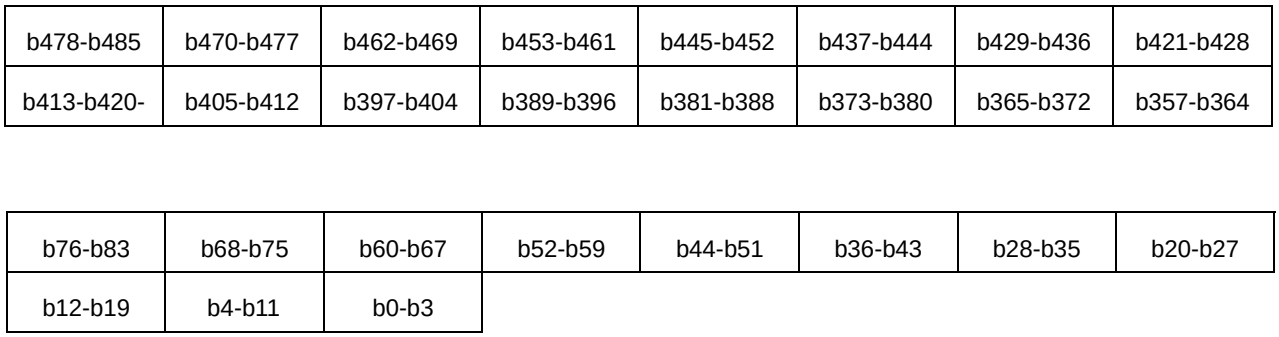


Figure 26-22. Descending Byte Ordering (DOSBY=0) of a 486 Hard Bits Output Block

26.3.2.1.4.2 Bit Ordering

The Hard Outputs bit order inside the byte is programmable. If the TVPEC0R[DOSBI] bit is set, then the Hard Outputs are arranged inside the bytes in ascending order as illustrated in **Figure 26-23**.

Figure 26-23. Ascending Bit Ordering (DOBSI=1) of Bits 222–229

If the TVPEC0R[DOSBI] bit is cleared, then the Hard Outputs are arranged inside the bytes in descending order as illustrated in **Figure 26-24**.

Figure 26-24. Descending Bit Ordering (DOBSI=0) of Bits 222–229

Note: The reset value of the TVPEC0R[DOSBI] and TVPEC0R[DOSBY] bits is *descending byte/bit ordering* (0), but the MAPLE-B changes this configuration during its initialization routine to *ascending byte/bit ordering* (1), which is its default configuration.

26.3.2.1.4.3 Byte/Bit Ordering Limitations

When working with Byte/Bit ordering other than the default configuration (DOSBI=DOSBY=1) the following must be considered:

- The DOSBI/DOSBY configuration must not be changed while there are active TVPE BDs in MAPLE-B. It is recommended to configure this register only once (after MAPLE-B activation), and leave it fixed for the whole operation duration.
- The CRC calculations (both, CRC stopping criteria and CRC on hard outputs) assume the default Byte/Bit ordering only (ascending byte/bit ordering). Working with different configuration will result in wrong CRC results!

26.3.2.1.5 TVPE Debug and Profiling

26.3.2.1.5.1 TVPE Debug

The MAPLE-B includes some debug capabilities which allow a host to view intermediate results of the TVPE. By allowing the MAPLE-B to output the TVPE internal extrinsic/Aposteriori results (using the SOE field in the TVPE BD), and by allowing it to limit the number of executed iterations (using the MAX_ITER field of the TVPE BD), it is possible to generate a flow by which the MAPLE-B outputs its extrinsic/Aposteriori results after every iteration.

For example, setting the SOE field to ‘10’ and the MAX_ITER to ‘0000’ will results in the TVPE executing one iteration only and on completion the MAPLE-B will output the extrinsic results from the TVPE internal memories to the system memory to the address described in the SFT_RSLT_ADDR field of the TVPE BD. If the extrinsic results of the second iteration are also of interest, it can be done by configuring the MAPLE-B with an additional BD, identical to the previous one, excluding the MAX_ITER field which should now be configured to ‘0001’ and with different SFT_RSLT_ADDR so the previous results are not overrun. By doing so, it is possible to get the extrinsic/Aposteriori results of every iteration.

26.3.2.1.5.2 TVPE Profiling

The MAPLE-B maintains the following counters which sum the following TVPE parameters:

- MTTPP (MAPLE-B Turbo Total Performance Parameter)—This parameter sums the total decoded bits of the TVPE Turbo decoding jobs. Once enabled, on every Turbo job completion it accumulates the recently completed job block size with its value, that is, MTTPP = MTTPP + BS.
- MVTPP (MAPLE-B Viterbi Total Performance Parameter)—This parameter sums the total decoded bits of the TVPE Viterbi decoding jobs. Once enabled, on every Viterbi job completion it accumulates the recently completed job block size with its value, that is, MVTPP = MVTPP + BS.
- MTBP (MAPLE-B Total BLER Parameter)—This parameter sums the total number of decoded blocks which have passed the CRC check. Once enabled, on every Turbo job completion, a CRC check is performed on the decoded block. If CRC passed, this parameter is increment by one.
- MTBDCP (MAPLE-B TVPE BDs Counter Parameter)—This parameter sums the total number of TVPE BDs completed by the MAPLE-B.

Enabling and disabling these counters is done using the profiling toggle bit ([PRF_TGL]) in the TVPE BDs. The first time the MAPLE-B finds this bit asserted in a TVPE BD, it starts the count of the enabled TVPE parameters (configured in **Section 26.4.2.3.9**, *MAPLE-B Profiling Enable Parameter (MPEP)*, on page 26-164). On the next time it finds this bit asserted in a TVPE BD, the MAPLE-B disables the count, thus allowing the host to read the values of the TVPE counters. On the following assertion of the [PRF_TGL] bit the MAPLE-B enables the count according to the MPEP enable bits, and so on.

Note: The BD which enables the count is counted in the profiling parameters while the BD which disables the count is not.

The host must initialize (or not) the counters parameters. The MAPLE-B (once enabled) performs only the following for the enabled parameters:

1. Read relevant counter value.
2. Add the required value (according to counter type).
3. Write back the counter value.

Note: The [PRF_TGL] bit of the TVPE BDs, can enable/disable only the TVPE profiling parameters, that is, the MTPPP, MVTPP, MTBP, and MTTBDCP only.

26.3.2.2 FFT/iFFT/DFT/iDFT Operation

The MAPLE-B supports FFT/iFFT/DFT/iDFT operations in variable block sizes as listed in **Section 26.1**, *Features*. These operations are executed using the FFTPE, for FFT/iFFT operation, and the DFTPE, for DFT/iDFT operation. Both PEs share the same programming model and the same operation flow, and hence are described together in this common section.

To simplify the explanations, the both the FFT and the DFT Processing Elements are referred henceforth in this section as FTPE.

Note: The FFT/iFFT/DFT/iDFT processing executed by the FFT/DFT Processing Elements are not standard related and therefore, are not affected by the MAPLE-B operation mode.

26.3.2.2.1 FTPE Buffer-Descriptor Structure

A description of the FTPE Buffer Descriptors structure expected by the host for the FFTPE/DFTPE BD rings is outlined in **Figure 26-24**. Some of the fields are status fields which are written by the MAPLE-B upon job completion. Other fields may be relevant only for certain configurations and will be ignored by the MAPLE-B if another configuration is used. The total length of the BD is 32 bytes, and its size does not depend on the configuration, that is, the next BD location should be located 32 bytes after the current BD address regardless of the BD configuration.

Offset BD_BASE + 0x0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWNER	WRAP	DC	INT_EN					TASK_ID							
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					PME	OVA_SCL	SCL_TYPE	ITE	GI		TL_ID					
W																

Figure 26-25. FTPE Buffer Descriptor Structure

Offset	BD_BASE + 0x4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IBA															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IBA													PRF_TGL		
W																

Offset	BD_BASE + 0x8															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBA															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBA															
W																

Offset	BD_BASE + 0xC															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PM_SCL	USR_SCL0				USR_SCL1				USR_SCL2				USR_SCL3		
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		USR_SCL4				USR_SCL5			IN_SCL				ADP_OVA_SCL			
W																

Offset	BD_BASE + 0x10															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R										BD_RPT						
W																

Offset	BD_BASE + 0x14															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BD_STAT_PTR															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BD_STAT_PTR															
W																

Figure 26-24. FTPE Buffer Descriptor Structure (Continued)

Offset	BD_BASE + 0x18															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		CMPLT_ST			ADP_OVA_SCL_ST											
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Offset	BD_BASE + 0x1C															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W																

Figure 26-24. FTPE Buffer Descriptor Structure (Continued)

Table 26-45. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Field	Description	Settings
OWNER 31	**Owner Bit** When set by the host, MAPLE-B is the BD owner and the job is not complete. When cleared by MAPLE-B, the job is done and the host is the BD owner.	0 The host is BD owner 1 MAPLE-B is BD owner
WRAP 30	**Wrap Bit** Indicates whether to use the sequential address (pointer + BD size) or to wrap to the BD base address for the next BD.	0 When cleared by the host, MAPLE-B expects to find the next BD at: M<pe>BR(H/L)PAxP[BDR_RD_PTR[13:3]] + BD size. 1 MAPLE-B updates M<pe>BR(H/L)PAxP[BDR_RD_PTR[13:3]] with M<pe>BR(H/L)PAxP[BDR_BA[13:8]], that is, it expects to find the next BD at the Base Address of the ring. **Note:** <pe> stands for FF or DF
DC 29	**Data Coherency.** Indicates whether the MAPLE-B should maintain data coherency when clearing the OWNER bit. If set, the MAPLE-B assures Data transfer completion before clearing the OWNER bit, else it clears the OWNER bit and shifts to the next BD regardless of Data transfer status. For details, see **Section 26.3.1.3**, *BDs Data Coherency Bit,* on page 26-10.	0 No Data coherency is required. 1 Data coherency is required. The MAPLE-B assures Data transfer completion before clearing the OWNER bit and shifting to the next BD.
INT_EN 28	**Interrupt Enable** If set, MAPLE-B issues an interrupt/doorbell interrupt at the end of this job. If cleared no interrupt is issued.	0 End of job interrupt is NOT issued. 1 End of job interrupt is issued.

Table 26-45. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Field	Description	Settings
— 27–24	Reserved	
TASK_ID[7:0] 23–16	**Task ID** Task ID tag, supplied by host, assists it in identification of its task when directly accessing the FTPE to inquire task status.	
— 15–12	Reserved	
PME 11	**Pre-Multiply Enable** Valid only during iFFT/FFT processing executed in the FFTPE. If set, the FFTPE will multiply each input sample with a complex value according the pre-multiply memory content. See **Section 26.3.2.2.2.4** and Section **Section 26.3.2.2.3.4.7** for details.	0 No action. 1 FFTPE multiplies each input sample with a complex value according the pre-multiply memory content.
OVA_SCL 10	**Overall Scaling** Valid only for adaptive scaling (SCL_TYPE==0). If set, the total scaling of the transform stages is aligned with the ADP_OVA_SCL field in the BD.	0 No overall scaling is specified. 1 The FTPE aligns the total accumulated scaling with the scaling value specified in the ADP_OVA_SCL filed in the BD.
SCL_TYPE 9	**Scaling Type** Selects the scaling method to be used between the transform stages as described in **Section 26.3.2.2.3.4**, *Scaling*.	0 Adaptive scaling is performed. 1 User defined scaling is performed.
ITE 8	**Inverse Transform Enable** If set, an inverse transform is executed.	0 FFT/DFT is executed. 1 iFFt/iDFT is executed.
GI 7	**Guard Insertion** Guard Insertion input data mode enable. Valid only during iFFT operation and must be cleared for any other operation. For more details see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.	0 Guard Insertion input data mode is disabled 1 Guard Insertion input data mode is enabled
— 6	Reserved	
TL_ID[5:0] 5–0	**Transform Length** This field determines the executed transform length. See **Table 26-53** for FFT field encoding and **Table 26-54** for DFT field encoding.	

Table 26-46. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Field	Description	Settings
IBA[31:3] 31–3	**Input Buffer Address** This field points to the input buffer location in system memory, where MAPLE-B is to fetch the data. The address must be aligned to an 8 byte address.	
PRF_TGL 2	**Profiling Toggle Bit** If set, the MAPLE-B changes its FFTPE/DFTPE profiling state from enabled/disabled to disabled/enabled. See **Section 26.3.2.2.6**, *FTPE Profiling,* on page 26-106	0 Do not toggle profiling state. 1 Toggle profiling state.
— 1–0	Reserved.	

Table 26-47. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Field	Description
OBA[31:3] 31–3	**Output Buffer Address** This field points to the output buffer location in system memory, where MAPLE-B is to write the result data. The address must be aligned to an 8 byte address.
— 2–0	Reserved.

Table 26-48. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Field	Description	
PM_SCL 31	**User Scaling after Pre-Multiplication** Valid only during iFFT/FFT processing executed in the FFTPE and only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of pre-multiplication.	0 no scaling is performed after the pre-multiplication. 1 the results of the pre-multiplication stage are scaled down (shift right) by 1.
USR_SCL0[2:0] 30–28	**User Scaling after Stage 0** Valid only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of stage 0.	000 no scaling is performed after stage 0 001 the results of stage 0 are scaled down (shift right) by 1. 010 the results of stage 0 are scaled down (shift right) by 2. 011 the results of stage 0 are scaled down (shift right) by 3. 100 the results of stage 0 are scaled down (shift right) by 4. 101 to 111 Reserved
— 27	Reserved	
USR_SCL1[2:0] 26–24	**User Scaling after Stage 1** Valid only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of stage 1.	000 no scaling is performed after stage 1 001 the results of stage 1 are scaled down (shift right) by 1. 010 the results of stage 1 are scaled down (shift right) by 2. 011 the results of stage 1 are scaled down (shift right) by 3. 100 the results of stage 1 are scaled down (shift right) by 4. 101 to 111 Reserved
— 23	Reserved	

Table 26-48. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC (Continued)

Field	Description	
USR_SCL2[2:0] 22–20	**User Scaling after Stage 2** Valid only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of stage 2.	000 no scaling is performed after stage 2 001 the results of stage 2 are scaled down (shift right) by 1. 010 the results of stage 2 are scaled down (shift right) by 2. 011 the results of stage 2 are scaled down (shift right) by 3. 100 the results of stage 2 are scaled down (shift right) by 4. 101 to 111 Reserved
— 19	Reserved	
USR_SCL3[2:0] 18–16	**User Scaling after Stage 3** Valid only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of stage 3.	000 no scaling is performed after stage 3 001 the results of stage 3 are scaled down (shift right) by 1. 010 the results of stage 3 are scaled down (shift right) by 2. 011 the results of stage 3 are scaled down (shift right) by 3. 100 the results of stage 3 are scaled down (shift right) by 4. 101 to 111 Reserved
— 15	Reserved	
USR_SCL4[2:0] 14–12	**User Scaling after Stage 4** Valid only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of stage 4.	000 no scaling is performed after stage 4 001 the results of stage 4 are scaled down (shift right) by 1. 010 the results of stage 4 are scaled down (shift right) by 2. 011 the results of stage 4 are scaled down (shift right) by 3. 100 the results of stage 4 are scaled down (shift right) by 4. 101 to 111 Reserved
— 11	Reserved	
USR_SCL5[2:0] 10–8	**User Scaling after Stage 5** Valid only if the SCL_TYPE field of the BD is set, that is, user defined scaling is chosen. Determines the scaling amount of the results of stage 5.	000 no scaling is performed after stage 5 001 the results of stage 5 are scaled down (shift right) by 1. 010 the results of stage 5 are scaled down (shift right) by 2. 011 the results of stage 5 are scaled down (shift right) by 4. 101 to 111 Reserved

Table 26-48. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC (Continued)

Field	Description	
IN_SCL[3:0] 7–4	**Input Scaling** Determines the up scaling of the input data before the processing of the first stage or before the pre-multiplication stage (if enabled).	0000 no scaling is performed on the input data before stage 0 0001 the input data is scaled up (shift left) by 1 before stage 0. 0010 the input data is scaled up (shift left) by 2 before stage 0. ... 1000 the input data is scaled up (shift left) by 8 before stage 0. 1001 to 1111 Reserved
ADP_OVA_SCL[3:0] 3–0	**Adaptive Overall Scaling** Valid only if Adaptive Scaling is chosen (SCL_TYPE==0), and the OVA_SCL field in the BD is set. Determines the Overall scale down the FTPE is to perform. The value of this field is compared with the total scaling accumulated by the adaptive scaling between the stages, and the output data is scaled up/down to fit the required overall scaling.	0000 The overall needed scaling is 0. 0001 The overall needed scaling is 1. 0010 The overall needed scaling is 2. ... 1011 The overall needed scaling is 11 1100 to 1111 Reserved

Table 26-49. FTPE Buffer Descriptor Fields Description of 14 Bytes at Offset 0x10

Field	Description	Settings
— 31–8	Reserved	
BD_RPT[7:0] 7–0	**BD Repeat** This field allows the use of the same BD, for multiple jobs with identical input parameters. For more details see **Section 26.3.2.2.1.1**, *BD Repeat Option*	0 BD repeat disabled. The BD is used once as normal BD 1 to 127 BD repeat enabled. This BD is used for 2 -128 jobs, respectively.

Table 26-50. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x14

Field	Description
BD_STAT_PTR[31:3] 31–3	**BD Status Pointer** This field points to the location in system memory where the MAPLE-B outputs the CMPLT_ST[2:0] and the ADP_OVA_SCL_ST[3:0] status fields of each of the jobs in the BD repeat option or of a single job in a non repeat mode. The pointer must be 8 byte aligned. Setting this field to 0x0000000 disables this feature.
— 2–0	Reserved

Table 26-51. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x18

Field	Description	Settings
CMP_RSN[2:0] 31–28	**Complete Status** This field describes the work completion status.	000 Task completed OK. 001 BD Config Error. Wrong TL_ID value. Job was not executed! 010 Task completed with saturation event.[1]
ADP_OVA_SCL_ST[3:0] 27–24	**Adaptive Overall Scaling Status** This field is valid only if Adaptive Scaling is enabled (SCL_TYPE = = 0). It describes the total Adaptive scaling applied during the processing stages[2].	0000 The total adaptive scaling applied is 0. 0001 The total adaptive scaling applied is 1. ... 1011 The total adaptive scaling applied is 11. 1100 to 1111 Reserved
— 23–0	Reserved	

1. If BD_RPT is enabled then the saturation indication for each Transform block in the repeat process is specified in the external memory pointer (if enabled), pointed by the BD_STAT_PTR field.
2. If BD_RPT is enabled the Adapting Overall Scaling indication for each Transform block in the repeat process is specified in the external memory pointer (if enabled), pointed by the BD_STAT_PTR field.

Table 26-52. FTPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x1C

Field	Description
— 31–0	Reserved

Table 26-53 and **Table 26-54** describe the transform length size encoding for the [TL_ID] field in the FTPE BDs.

Table 26-53. [TL_ID] Field Encoding for the Different Transform Lengths of the FFTPE BD

Transform Length	[TL_ID] Field of FFT BD
128	0
256	1
512	2
1024	3
2048	4

Table 26-54. [TL_ID] Field Encoding for the Different Transform Lengths of the DFTPE BD

Transform Length	[TL_ID] Field of DFT BD	Transform Length	[TL_ID] Field of DFT BD	Transform Length	[TL_ID] Field of DFT BD
768	26	240	13	12	0
864	27	288	14	24	1
900	28	300	15	36	2
960	29	324	16	48	3
972	30	360	17	60	4
1080	31	384	18	72	5

Table 26-54. [TL_ID] Field Encoding for the Different Transform Lengths of the DFTPE BD

Transform Length	[TL_ID] Field of DFT BD	Transform Length	[TL_ID] Field of DFT BD	Transform Length	[TL_ID] Field of DFT BD
1152	32	432	19	96	6
1200	33	480	20	108	7
1536	34	540	21	120	8
128[1]	35	576	22	144	9
256[1]	36	600	23	180	10
512[1]	37	648	24	192	11
1024[1]	38	720	25	216	12

1. FFT executed on the DFTPE.

26.3.2.2.1.1 BD Repeat Option

The MAPLE-B allows the user to generate a single FTPE Buffer Descriptor to describe up to 128 different FFT or DFT jobs for the MAPLE-B to execute. To work in this mode, several conditions must be met:

1. All the jobs must share the following parameters:
   - SCL_TYPE. Scaling Type (User define or Adaptive scaling. See **Section 26.3.2.2.3.4**, *Scaling*)
   - OVA_SCL. Overall Scaling (enabled or disabled. See **Section 26.3.2.2.3.4**, *Scaling*)
   - ITE - Inverse Transform Enable. See **Section 26.3.2.2.3.2**, *Inverse Transform Processing*
   - TL_ID. Transform Length ID.
   - USR_SCLx. User Defined Scaling for stage 0 to stage 5. See **Section 26.3.2.2.3.4**, *Scaling*
   - IN_SCL. Input Scaling. See **Section 26.3.2.2.3.4**, *Scaling*
   - ADP_OVA_SCL. Adaptive Overall Scaling. See **Section 26.3.2.2.3.4**, *Scaling*
2. All the input data for the jobs must be located in system memory sequentially, starting from IBA (Input Buffer Address), as described in **Figure 26-27**.
3. The BD_RPT field in the BD must be initialized with value range 1 to 127 to execute 2 to 128 jobs, respectively.

If all the above conditions are met, when the MAPLE-B starts the BD execution, it will sequentially execute all the jobs. The MAPLE-B will write the output data from the jobs sequentially into system memory, starting at the address described in the OBA (Output Buffer Address).

The Owner bit is cleared by the MAPLE-B after all job executions are done.

If the INT_EN (Interrupt Enable) bit is set in the BD, an interrupt is issued only after all the job executions of the BD are complete.

To save the status fields of each job processed during the BD repeat option, the MAPLE-B uses the [BD_STAT_PTR] field as a pointer into system memory where it copies the two status fields of each job (CMPLT_ST[2:0] and ADP_OVA_SCL_ST[3:0]).

Figure 26-25 describes the data structure of the status fields in system memory:

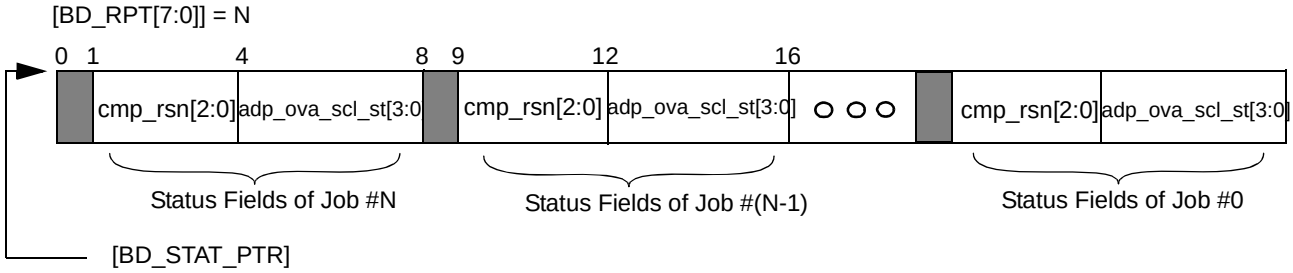


Figure 26-25. Status Table Data Structure in System Memory in Case of BD Repeat

For details on the ADP_OVA_SCL_ST field see **Section 26.3.2.2.3.4**, *Scaling*.

For details on the CMP_RSN field see **Table 26-51**

Note: The repeat option is highly recommended for use in small Transform blocks (< 128) since it reduces the overhead of finding and parsing the following BD, assuming the same job type is required.

Note: If the TL_ID value is wrong (CMP_RSN[2:0]=001) in case of BD repeat option is enabled, the MAPLE-B will not output the status fields described in <Cross Refs>Figure 26-25 to the system memory, but will indicate it in the BD status fields.

Note: It is possible to disable the status fields update to the system memory by resetting the [BD_STAT_PTR] field in the BD (BD_STAT_PTR=0x0). Doing so may slightly improve the throughput performance of the DFTPE/FFTPE.

Note: The [BD_STAT_PTR] field is relevant for non repeat jobs as well. If enabled ([BD_STAT_PTR] > 0), the MAPLE-B will output the status fields regardless of the repeat option.

26.3.2.2.1.2 Buffer Descriptors Notes

- Once the [OWNER] bit is set by the host, the BD content must not be changed. It can be re-written only after the [OWNER] bit was cleared by MAPLE-B.
- The Input Data Address (IBA) and Output Data Address (OBA) fields in the BD must be aligned to 8 bytes.
- All the status fields in the BD are updated by the MAPLE-B once the job is complete, that is, they are valid only after the owner bit is cleared by the MAPLE-B.
- The [TL_ID] valid value are specified in **Table 26-53** and **Table 26-54** only. Other values result in an MAPLE-B interrupt (if not masked).
- When a host is writing a new BD to its ring, it must be done “bottom-up”, that is, the [OWNER] bit must be the last write access to the PSIF DRAM.

26.3.2.2.2 FTPE Data Structures

The input data to FTPE is fetched by the MAPLE-B internal DMA. The location of the input data in system memory should be specified in the IBA[31:3] field of the FTPE BD. Upon job completion, the output data of the FTPE is transferred to external memory by the MAPLE-B internal DMA. The address to transfer the data is specified in the OBA[31:3] field of the FTPE BD.

The following sections describe the expected input data structure and MAPLE-B output data structure in system memory.

26.3.2.2.2.1 Input Data Structure

After parsing the FTPE BD, MAPLE-B starts fetching the input data into its internal memories for the processing operation. The FTPE input (and output) data is composed of 32 bit samples, each containing a 16 bit real component of the sample, followed by a 16 bit imaginary component of the sample. **Table 26-55** describes the data structure of two 32 bit samples arriving from system memory:

Table 26-55. Two 32 Bits Samples as They are Expected to Arrive From System Memory

Bits	63:48		47:32		31:16		15:0	
Bytes	0	1	2	3	4	5	6	7
Samples	Sample k				Sample k+1			
Value	Real value		Imaginary value		Real value		Imaginary value	

Figure 26-26 shows an example of a DFT block of Transform Length 12, as it should be placed in a 128 bit memory structure. The base address of the example buffer is 0x18:

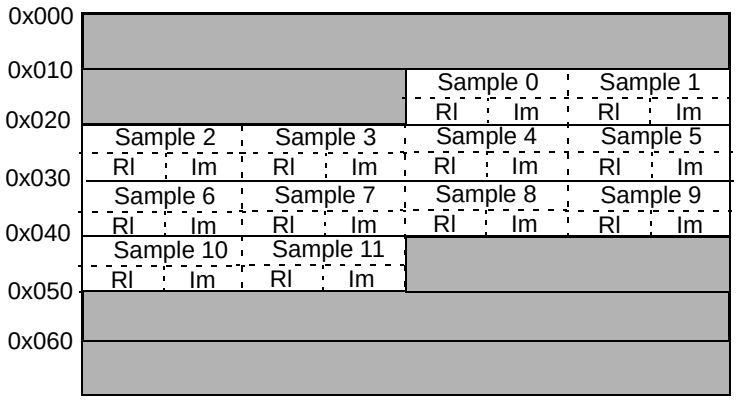


Figure 26-26. DFT Block, Transform Length 12, Placed in 128-Bit Memory, Start Address 0x18

Note: The Input Buffer Address in the external memory must be aligned to a 64 bit address, as shown in the example in **Figure 26-26**.

If the BD_RPT field in the FTPE BD is enabled (BD_RPT>0), then the Input Buffer Address should point to the location in system memory, where all the jobs related to that BD are located successively. **Figure 26-27** illustrates an example of how the data should be arranged in a 128 bit wide system memory if 5 jobs of Transform Length 12 are assigned to one BD:

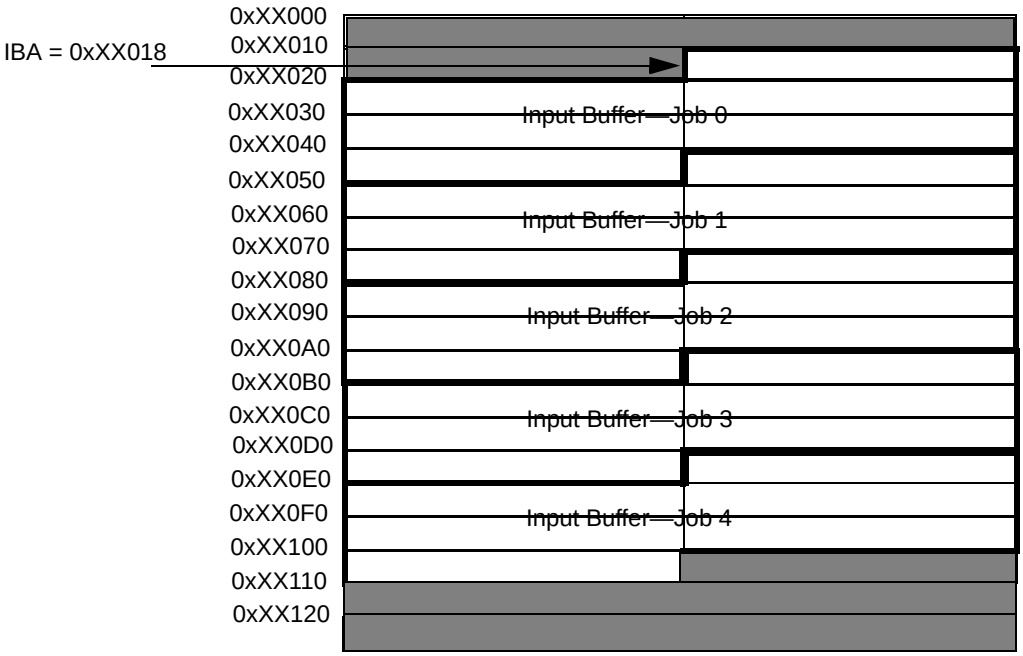


Figure 26-27. Input Buffer Structure for BD_RPT= 4 and Transform Length = 12

26.3.2.2.2.2 Guard Band Insertion for iFFT

To reduce system loading, the FTPE supports a special input data structure for iFFT processing only. This special input data structure allows the MAPLE-B to fetch only the relevant parts of the input data, and the FTPE can internally generate the whole input data block structure.

Figure 26-28 is an example of the internal structure in the frequency domain of a 1024 sample iFFT symbol:

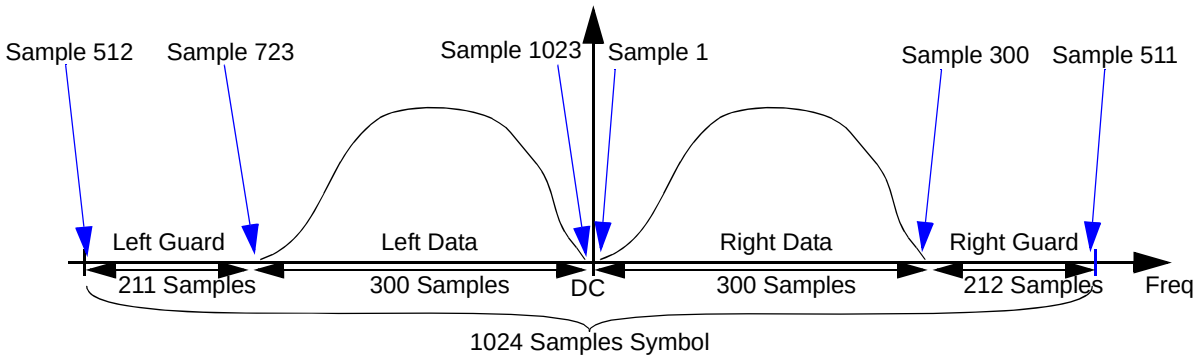


Figure 26-28. 1024 Samples Frequency Domain Symbol Example

As seen in **Figure 26-28**, the symbol is built from the following elements:

- Left data and right data which are equal in size
- Left guard and right guard
- One sample of DC

The Right/Left guards as well as the DC sample are all zero value samples.

When the Info Sub-carrier bit ([ISC] bit in the FTPE BD) is enabled, the MAPLE-B fetches only the Left-data following by the Right-data from system memory, and the FTPE internally generates the whole symbol structure as is seen in **Table 26-29**.

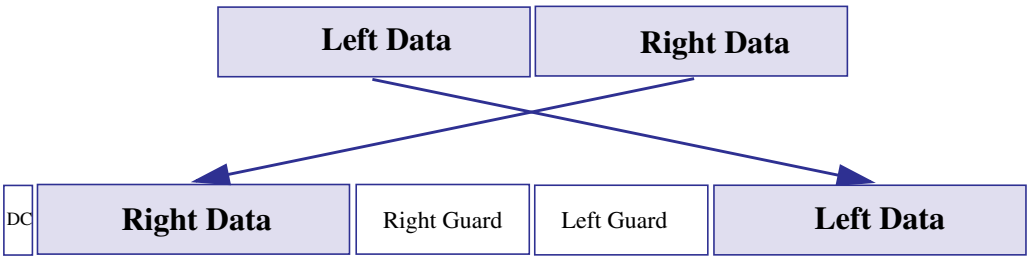


Figure 26-29. Generating Whole iFFT Symbol Using the Right/Left Data

The size of the Right/Left data for each of the possible transform lengths must be specified in the FTPE FTPEDSRx registers. **Table 26-56** describes which fields in these registers relate to which Transform Length, and also the boot value for each of the fields with respect to the standard.

Table 26-56. FTPEDSRx Fields and the Related Transform Length

Register [Field]	Transform Length	Reset Value for WiMAX[1]	Reset Value for 3GLTE[1]
FTPEDSR0[DS0]	128	42	36
FTPEDSR0[DS1]	256	NA	90
FTPEDSR1[DS2]	512	210	150
FTPEDSR1[DS3]	1024	420	300
DFTPEDSR2[DS4][2]	1536	NA	450
FFTPEDSR2[DS4][3]	2048	840	600

1. The values in **Table 26-56** equals FTPEDSRx[DSy] * 2
2. The 1536 Transform Length is supported only in the DFTPE.
3. The 2048 Transform Length is supported only in the FFTPE.

If needed, the host can change these values by writing to these registers.

Note: The [GI] bit can be used only while processing an iFFT job. It must be cleared for any other operation.

The FTPE derives the size of the DC + Left Guard + Right Guard (DC + LG + RG), which is the number of zero samples it needs to generate, according to the following equation:

$$DC + LG + RG = Transform\ Length - 4 \times FTPEDSRx[DSy]$$

where the *x* and *y* of the FTPEDSR*x*[DS*y*] are derived from the TL value and according to **Table 26-56**.

26.3.2.2.2.3 Output Data Structure.

Once the FTPE has completed its processing, it outputs the processed data to system memory. The address it uses is specified in the [OBA] field of the BD.

The output data structure of the FTPE is identical to the input data structure, that is, it is composed of 32 bit samples as described in **Table 26-55**, and it is written to system memory with the same structure as the input data buffer (see **Figure 26-26**).

If BD_RPT is enabled (BD_RPT>0), the MAPLE-B outputs the data of all the related jobs to successive addresses starting at the OBA (Output Buffer Address). The structure of the output data is arranged successively in memory and is similar to the structure of the input data for this case. (see example in **Figure 26-27**).

26.3.2.2.2.4 Pre-Multiplication Support for FFT/iFFT Processing in the FFTPE

The FFTPE includes a pre-multiplication unit that allows multiplying each of the FFT input samples with a different complex number represented by a 16-bit fractional fixed point representation of 1q15 for the real part and for the imaginary part. These complex numbers

should be written into the pre-multiplication memory after MAPLE-B initialization or while the FFTPE is in idle state.

Note: The pre-multiplication memory is ECC-protected; therefore, writing into this memory must be executed with full bus width access (64 bits).

Enabling the pre-multiplication feature in the FTPE is done by asserting the [PME] bit in the FFTPEBD. The multiplications of the complex numbers in the pre-multiplication memory is always executed according to the order of the input samples stream. That is, the first input sample is multiplied by the complex number located in address 0x0 of the pre-multiplication memory; the second input sample is multiplied by the complex number located in address 0x4 of the pre-multiplication memory, and so forth.

The pre-multiplication memory size is 8 Kbyte in order to fit the maximum FFT size of 2048 (4 bytes for each complex number). Consequently, for smaller FFT sizes, the pre-multiplication memory is not fully utilized.

Figure 26-30 describes the pre-multiplication memory utilization for the different FFT sizes and the order of multiplications of its complex numbers.

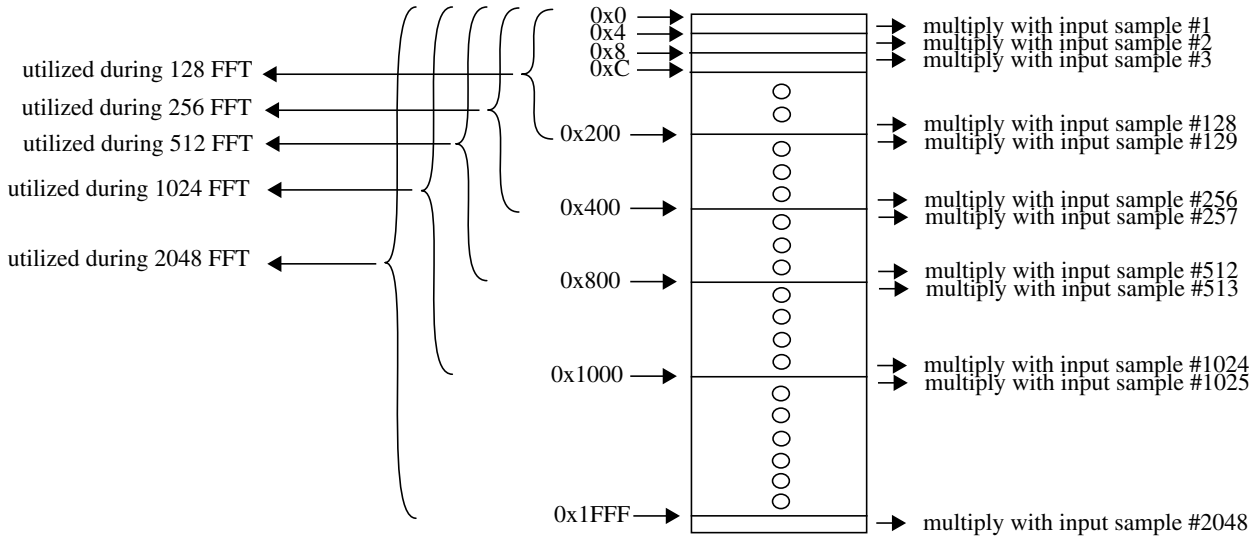


Figure 26-30. Pre-Multiplication Memory Utilization for the Different FFT Sizes

Note: If the [GI] bit of the FFTPE BD is asserted during iFFT processing, the expected input data is different (see Guard Band Insertion for iFFT). In that case, if the pre-multiplication option is enabled, the content (order and amount of complex numbers) of the pre-multiplication memory should be adjusted according to the new input data order.

Note: If the pre-multiplication memory is initialized with complex numbers for a certain Transform Length (TL) and a different TL size is required for processing, the FFTPE must be in idle in order to change the content of the pre-multiplication memory.

The pre-multiplication operation affects the scaling scheme of the FFT/iFFT processing in the FFTPE. For more details on the pre-multiplication scaling, see Pre-multiplication scaling. The pre-multiplier operation can be used to eliminate the half sub-carrier shift, in FFT processing of the UL SC-FDMA signals as described in 3GPP TS

26.3.2.2.3 FTPE Processing

26.3.2.2.3.1 FTPE Algorithm

The FFT/DFT transformation is calculated using a mixed Radix algorithm. This algorithm is based on DIF (Decimation In Frequency) partitioning of the transform length into 2 or more operands whose multiplication equals the transform length. For example, a transform length of 24 can be partitioned to 3 calculation stages, using radices 4, 3 and 2($24 = 4 \times 3 \times 2$).

The following figures show an example of Mixed Radix DIF (Decimation in Frequency) partitioning of a transform length 24 into three stages. During the first stage, the FTPE executes DFT-4 using its Radix-4 module; during the second stage the FTPE executes DFT-3 using its Radix-3 module; and in the third and last stage, it executes DFT-2 using its Radix-2 module. **Figure 26-31** describes the mathematics of stage 0 partitioning into DFT-4 and DFT-6:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi nk}{N}} \qquad k = 0, \ldots, N-1$$

$$X(k) = \sum_{n=0}^{23} x(n) \cdot e^{-j\frac{2\pi nk}{24}} = \sum_{n=0}^{5} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=6}^{11} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=12}^{17} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=18}^{23} x(n) \cdot e^{-j\frac{2\pi nk}{24}}$$

$$= \sum_{n=0}^{5} x(n) \cdot e^{-j\frac{2\pi nk}{24}} + \sum_{n=0}^{5} x(n+6) \cdot e^{-j\frac{2\pi nk}{24}} \cdot e^{-j\frac{\pi k}{2}} + \sum_{n=0}^{5} x(n+12) \cdot e^{-j\frac{2\pi nk}{24}} \cdot e^{-j\pi k} + \sum_{n=0}^{5} x(n+18) \cdot e^{-j\frac{2\pi nk}{24}} \cdot e^{-j\frac{3\pi k}{2}}$$

$$= \sum_{n=0}^{5} \{x(n) + (-j)^k x(n+6) + (-1)^k x(n+12) + (j)^k x(n+18)\} \cdot e^{-j\frac{2\pi nk}{24}}$$

$$X(4k) = \sum_{n=0}^{5} \{x(n) + x(n+6) + x(n+12) + x(n+18)\} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(4k+1) = \sum_{n=0}^{5} [x(n) - jx(n+6) - x(n+12) + jx(n+18)] \cdot e^{-j\frac{2\pi n}{24}} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(4k+2) = \sum_{n=0}^{5} [x(n) - x(n+6) + x(n+12) - x(n+18)] \cdot e^{-j\frac{4\pi n}{24}} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(4k+3) = \sum_{n=0}^{5} [x(n) + jx(n+6) - x(n+12) - jx(n+18)] \cdot e^{-j\frac{6\pi n}{24}} \cdot e^{-j\frac{2\pi nk}{6}}$$

DFT 6 DFT 4 Twiddles DFT 6

Figure 26-31. Example Mixed Radix Partitioning of Transform Length 24 (First Stage Mathematics)

Figure 26-32 shows the first stage partitioning implementation:

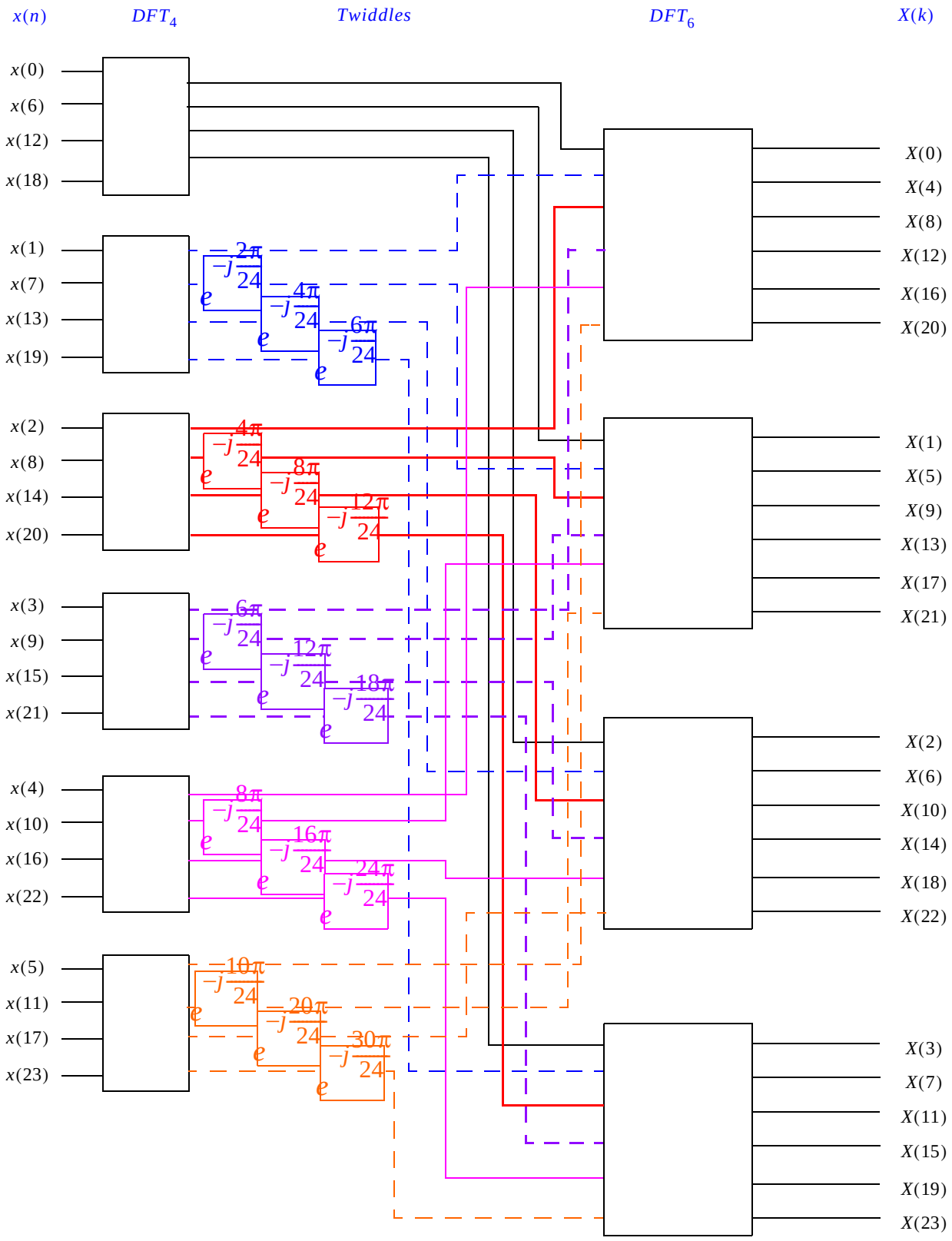


Figure 26-32. Example Mixed Radix Partitioning of Transform Length 24 (First Stage Implementation)

Figure 26-33 describes the second and third stage partitioning into DFT-3 and DFT-2, which are executed using the Radix-3 and Radix-2 modules, respectively

$$X(k) = \sum_{n=0}^{5} x(n)\cdot e^{-j\frac{2\pi nk}{6}} = \sum_{n=0}^{1} x(n)\cdot e^{-j\frac{2\pi nk}{6}} + \sum_{n=2}^{3} x(n)\cdot e^{-j\frac{2\pi nk}{6}} + \sum_{n=4}^{5} x(n)\cdot e^{-j\frac{2\pi nk}{6}}$$

$$= \sum_{n=0}^{1} x(n)\cdot e^{-j\frac{2\pi nk}{6}} + \sum_{n=0}^{1} x(n+2)\cdot e^{-j\frac{2\pi nk}{6}}\cdot e^{-j\frac{4\pi k}{6}} + \sum_{n=0}^{1} x(n+4)\cdot e^{-j\frac{2\pi nk}{6}}\cdot e^{-j\frac{8\pi k}{6}}$$

$$= \sum_{n=0}^{1} \left\{ x(n) + x(n+2)\cdot e^{-j\frac{2\pi k}{3}} + x(n+4)\cdot e^{-j\frac{4\pi k}{3}} \right\} \cdot e^{-j\frac{2\pi nk}{6}}$$

$$X(3k) = \sum_{n=0}^{1} \{x(n) + x(n+2) + x(n+4)\}\cdot(-1)^{nk}$$

$$X(3k+1) = \sum_{n=0}^{1} \left\{ \left[x(n) + x(n+2)\cdot e^{-j\frac{2\pi}{3}} + x(n+4)\cdot e^{-j\frac{4\pi}{3}} \right] \cdot e^{-j\frac{2\pi n}{6}} \right\} \cdot (-1)^{nk}$$

$$X(3k+2) = \sum_{n=0}^{1} \left\{ \left[x(n) + x(n+2)\cdot e^{-j\frac{4\pi}{3}} + x(n+4)\cdot e^{-j\frac{8\pi}{3}} \right] \cdot e^{-j\frac{4\pi n}{6}} \right\} \cdot (-1)^{nk}$$

DFT 2 DFT 3 Twiddles DFT 2

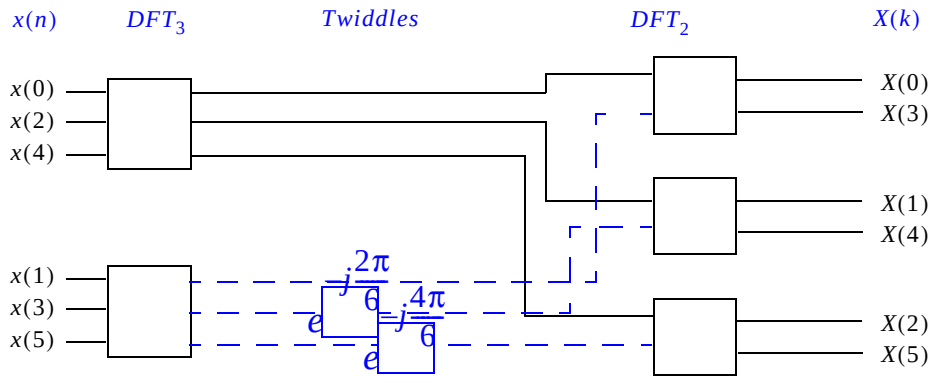


Figure 26-33. Example Mixed Radix Partitioning of Transform Length 24 (Second and Third Stage Mathematics)

26.3.2.2.3.2 Inverse Transform Processing

The FTPE can calculate an Inverse transform. Calculating an Inverse Fourier transform is identical to calculating a Fourier transform except that in the Inverse Fourier transform the multiplication factor is $e^{j\frac{2\pi nk}{N}}$ instead of $e^{-j\frac{2\pi nk}{N}}$ as in the Fourier transform.

To calculate the Inverse Fourier transform, the ITE (Inverse Transform Enable) bit in the FTPE BD must be set.

26.3.2.2.3.3 Transform Length Partitioning

For each possible DFT/iDFT/FFT/iFFT transform length (as listed in **Section 26.1**, *Features*) the FTPE has a fixed partitioning algorithm, which determines the number of partitioning stages, the order of the stage execution, and the Radices used for each stage.

1. The implemented Radices in the DFTPE are: Radix- 5, Radix- 4, Radix- 3 and Radix- 2.
2. The implemented Radices in the FFTPE are: Radix- 8, Radix- 4 and Radix- 2.
3. The Transform Length of each job is determined in the TL_ID field of the FTPE BD.

Table 26-57 describes the partitioning of all the possible DFT transform lengths of the DFTPE. Each stage column describes the executed Radix for that stage.

Table 26-57. Partitioning Path for Supported DFTPE Transforms

Transform Length	Stage 0	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
12	4	3				
24	4	3	2			
36	4	3	3			
48	4	4	3			
60	5	4	3			
72	4	3	3	2		
96	4	4	3	2		
108	4	3	3	3		
120	5	4	3	2		
144	4	4	3	3		
180	5	4	3	3		
192	4	4	4	3		
216	4	3	3	3	2	
240	5	4	4	3		
288	4	4	3	3	2	
300	5	5	4	3		

Table 26-57. Partitioning Path for Supported DFTPE Transforms (Continued)

Transform Length	Stage 0	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
324	4	3	3	3	3	
360	5	4	3	3	2	
384	4	4	4	3	2	
432	4	4	3	3	3	
480	5	4	4	3	2	
540	5	4	3	3	3	
576	4	4	4	3	3	
600	5	5	4	3	2	
648	4	3	3	3	3	2
720	5	4	4	3	3	
768	4	4	4	4	3	
864	4	4	3	3	3	2
900	5	5	4	3	3	
960	5	4	4	4	3	
972	4	3	3	3	3	3
1080	5	4	3	3	3	2
1152	4	4	4	3	3	2
1200	5	5	4	4	3	
1536	4	4	4	4	3	2
FFT Support in DFTPE Block[1]						
128	4	4	4	2		
256	4	4	4	4		
512	4	4	4	4	2	
1024	4	4	4	4	4	

1. It is possible to use the DFTPE as an FFT/iFFT processor. For more details, see **Section 26.3.2.2.5**, *Using the DFTPE as FFTPE*, on page 26-105.

Table 26-58 describes the partitioning of all the possible FFT transform lengths of the FFTPE. Each stage column describe the executed Radix for that stage.

Table 26-58. Partitioning Path for Supported FFTPE Transforms

Transform Length	Stage 0	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
128	8	8	2			
256	8	8	4			
512	8	8	8			
1024	8	8	8	2		
2048	8	8	8	4		

26.3.2.2.3.4 Scaling

The FTPE supports two types of down scaling methods, which are applied in between the calculation stages and are targeted to reduce or eliminate the possibility of samples reaching saturation. The scaling methods are:

- Adaptive Scaling
- User Defined Scaling

Along with these two scaling methods, which are applied during the whole transform processing, it is possible to apply up-scaling on the input data at the beginning of the processing. For details, see **Section 26.3.2.2.3.4**, *Scaling*.

In the FTPE implementation, each Radix module inputs a different number of bits. The number of input bits to each Radix depend on the maximum bit expansion of the Radix. The output number of bits of all the Radix stages is 20, with the Radix point location of 5q15[1]. The input number of bits to each Radix stage is described in **Table 26-59**.

Table 26-59. Input Bits per Radix

Radix	Input Bits	Radix Point Location
2	18	3q15
3	17	2q15
4	17	2q15
5	16	1q15
8	16	1q15

26.3.2.2.3.4.1 Input Scaling

This feature is relevant for input data to the FTPE that is characterized as small value samples which, with additional scale up, will result in more precise output results.

The FTPE can be programed to scale up the input data up to 8 bits. This is done by configuring the IN_SCL[3:0] field of the FTPE BD with values between 1 to 8. The value of 0 means no input scaling is performed.

If saturation occurs due to input scaling, the FTPE will use the largest/smallest number possible (sign dependent), instead of the overflown sample.

Input scaling can only up scale (shift left) the input data.

1. The (x)q(y) representation indicates the location of the mathematical point where all the bits to its right represent the fraction and all the bits to it left represent the integer. For example 2q15 means 2 bits left of the mathematical point (one sign bit and one integer bit) and 15 fractional bits.

26.3.2.2.3.4.2 User Defined Scaling

Given the information on number of input and output bits for each Radix (**Table 26-59**), and the Radix stage order for each transform length (**Table 26-57** and **Table 26-58**), the FTPE can be programed to execute down scaling after each stage according to user programming, to prevent samples saturation.

To enable User Defined Scaling, the SCL_TYPE bit in the FTPE BD must be set. Using the USR_SCLx[2:0] (x = 0... 5) fields of the FTPE BD, the FTPE can be programmed to execute scaling down after each of the possible stages. For example, by assigning a value to USR_SCL0[2:0], the FTPE will perform down scaling on the output of the Radix which was applied during stage 0 with the scaling factor in the USR_SCL0[2:0].

The USR_SCLx[2:0] fields can be programmed with values from 0 to 4as the amount of down scaling to be performed. The User Defined Scaling supports only down scaling.

26.3.2.2.3.4.3 Saturation

When using User Defined Scaling, some of the data samples may overflow due to the Radix calculation. For overflown samples, the FTPE uses the largest/smallest number possible (depending on the sign bit) instead of the overflown sample. The FTPE also maintains counters for the number of overflown samples for each Radix stage. These counters are implemented as FTPE Status registers. For more details see **Section 26.3.2.2.4**, *FTPE Status Indications*."

26.3.2.2.3.4.4 Adaptive Scaling

In Adaptive Scaling mode the, the FTPE performs independent down scaling after each Radix stage if needed. The amount of scaling applied after each Radix stage is the minimum down scaling necessary to prevent saturation of any of the samples during the next Radix calculation.

To enable Adaptive Scaling, the SCL_TYPE bit of the FTPE BD must be cleared.

The FTPESCLSTR[SCLx] status fields contain the amount of down scaling performed by the FTPE during each processing stage. They are valid for reading after every job completion.

The total amount of down scaling performed by the FTPE during transform processing using Adaptive Scaling mode is recorded in the ADP_OVA_SCL_ST[3:0] field of the FTPE BD.

If BD_RPT is enabled, which means more than one job for a single BD (see **Section 26.3.2.2.1.1**, *BD Repeat Option*), the total overall scaling amount for all the jobs is transferred to system memory to the address pointed to by the ADP_OVA_SCL_PTR field in the FTPE BD.

Note: It may happen that during adaptive scaling operation the saturation status indication in the FTPE BD is set ([CMP_RSN] = 0x2). This can be caused due to internal FTPE rounding issues and does not indicate that actual saturation has occur, therefore should be ignored.

26.3.2.2.3.4.5 Overall Scaling Amount Programming

The host can define the overall amount of needed scaling to be applied during the transform processing when using Adaptive scaling mode. If enabled, the FTPE adjusts its accumulated scaling amount to the Overall Scaling factor programmed by the user. The scaling the FTPE applies at the end of the transform to fit the overall scaling factor can be either up-scaling or down-scaling. For example, if the accumulated adaptive scaling of a given transform is 6(down scaling), and the required Overall Scaling is 4, the FTPE will up-scale the final results by 2. If the required Overall Scaling is 8, the FTPE will down-scale the final results by 2. The required Overall Scaling amount calculation does not take into account the input scaling (if applied), but only the accumulated adaptive down-scaling during stage processing.

To enable the Overall Scaling factor, the OVA_SCL bit in the FTPE BD must be set, and the ADP_OVA_SCL[3:0] field in the FTPE BD, must contain the Overall Scaling factor.

If Overall Scaling is enabled, the ADP_OVA_SCL_ST[3:0]/ADP_OVA_SCL_Jx_ST[3:0] status indications in the FTPE BD is equal to the ADP_OVA_SCL field of the FTPE BD, since the later determines the overall scaling that is applied during the transform processing.

Note: The overall scaling adjustment in the FTPE is executed in the last stage of the transform. The FTPESCLSTR[SCLx] field of the last stage holds the required adaptive scaling for this stage without including the Overall Scaling correction that may occur due to enabling the Overall Adaptive Scaling. It means that the actual value of the FTPESCLSTR[SCLx] for the last stage may not be correct if overall scaling is enabled.

26.3.2.2.3.4.6 Adaptive Scaling Example

Figure 26-34 shows an example of data flow through the Radices and scaling stages with respect to the number of bits at the input to each Radix stage. The execution of the example is explained for each stage:

1. 16 bits of data from the input buffer are scaled up by 5 according to the input scaling configuration.
2. The 16 bits after input scaling are sign-extended to 17 bits as required for the Radix 3 stage input.
3. The output of the Radix 3 is a 20-bit sample.
4. Because the next stage performs Radix 2, the output sample of the current Radix (Radix 3) are scaled down by 2 bits to fit the Radix 2 input size (18 bits).
5. In order to fit the 16 bits output requirement, the 20 bit results of the Radix 2 as described in **Figure 26-34** are scaled down by 1 bit and the 3 msb are removed (because the 4 msb are identical, the FTPE considers this a sign extension).

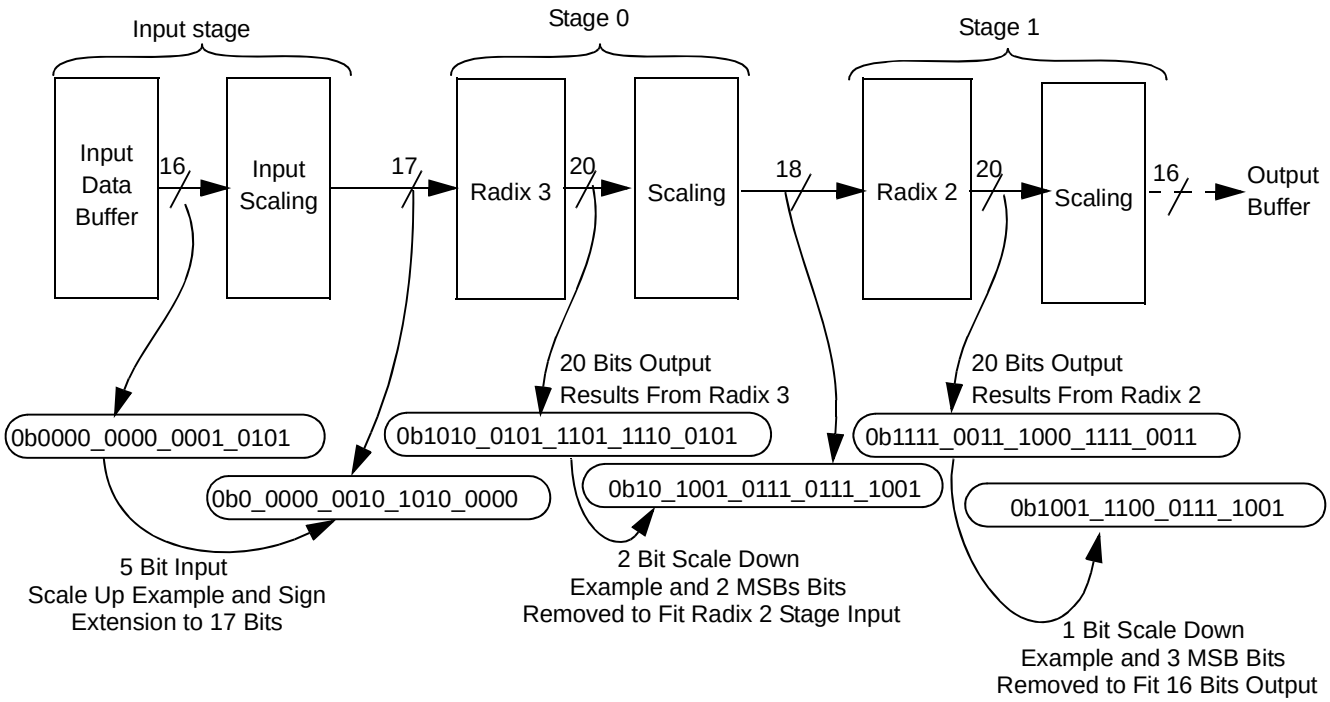


Figure 26-34. Input Scaling Example and Stage Scaling Example (Adaptive or User Defined)

Note: The previous example describes a partitioning of transform length of 6 (3×2), which is not a supported transform length of the FTPE. The example is for explanation only.

26.3.2.2.3.4.7 Pre-Multiplication Scaling

If the pre-multiplication option is enabled in the FFTPE (by setting the [PME] bit in the FFTPEBD), the scaling result of the FFTPE processing may be affected as follows:

- Because the pre-multiplication complex numbers are represented by a 16-bit fractional fixed point representation (1q15) for both the real part and for the imaginary part, multiplying these complex numbers with the input samples of the FFTPE may cause the result to grow up to a magnitude of 2, thus increase the scaling of the result by up to 1.
- If adaptive scaling is enabled ([SCL_TYPE] bit of the FFTPE BD is cleared), the FFTPE internally performs the scaling (if required) and if scaling is performed after the pre-multiplication stage, it will be added to the overall scaling status reported by the [ADP_OVA_SCL_ST] field of the FFTPE BD.
- If user scaling is enabled ([SCL_TYPE] bit of the FFTPE BD is set), the FFTPE uses the [PM_SCL] bit in the FFTPE BD in order to determine if scaling is performed after the pre-multiplication stage. If the [PM_SCL] bit is set, the result of the pre-multiplication stage will be scaled down by one. The number of saturated samples (if they exist) caused by this scaling can be viewed in the FFTPESSTR0[SAT0] field of the FFTPE registers.

26.3.2.2.4 FTPE Status Indications

The FTPE has the following status indicators, some of which are read directly from the FTPE status registers, and some are read from the FTPE BD after job completion.

- FTPESTR[1][TASKIDS]—This field is copied from the TASK_ID field of the FTPE BD, to indicate which BD the FTPE is currently processing. This field is updated whenever a new BD is processed.
- FTPESCLSTR[SCLx]—This field describes the scaling factor applied in each stage during Adaptive Scaling mode.
- FTPESCLSTR[OVA_SCL]—This field describes the overall scaling applied during Adaptive scaling mode. This field is also copied by the MAPLE-B into the *FTPE BD[ADP_OVA_SCL_ST]*.
- FTPESSTRx[SATy]—This field contains the saturation counters for each stage. For each stage, the counters show the number of saturated samples. These status fields are valid only during User Scaling mode.
- BD[2][CMP_RSN]—This field describes the status of the completed job. It indicates whether the job completed OK, completed with error due to wrong TL_ID value, or completed with saturation events during the processing.
- BD[ADT_OVA_SCL_ST]—This field is copied by the MAPLE-B from the *FTPESCLSTR[OVA_SCL]* status register.

Note: All the FTPExSTRs are cleared on every new job the FTPE starts. To be able to read these fields, the FTPE must receive no new jobs after the current job. This gives the host the needed time to access these registers and read the relevant fields.

26.3.2.2.5 Using the DFTPE as FFTPE

Due to the similar implementation of DFTPE to the FFTPE, it is possible to use the DFTPE to execute FFT/iFFT processing. This is done by placing an FFT/iFFT Transform Length size (128, 256, 512, or 1024[3]) in any of the DFTPE BDs. The DFTPE is designed to perform FFT processing automatically on any of the above FFT/iFFT Transform Length sizes. All FFTPE features (including [ISC] capabilities) are supported by the DFTPE as well.

Executing FFT processing in the DFTPE has a few limitations which must be considered:

- Transform length size. Due to internal memory limitation the DFTPE can not perform a 2048 FFT transform length.

1. All the FTPE... registers refer to the FFTPE... and DFTPE... registers as described in **Section 26.4.4.4.3**, *DFTPE Saturation Status Register 0 (DFTPESSTR0)* and **Section 26.4.4.4.1**, *DFTPE Status Register (DFTPESTR)*."
2. All the BD[xxx] fields refer to the FTPE BD fields.
3. The DFTPE can not perform a 2048 FFT Transform Length due to internal memory limitation

- Throughput. Due to the DFTPE internal implementation (supporting variable block sizes for the DFT), the total throughput of the DFTPE while performing FFT transforms is lower in most cases.

26.3.2.2.6 FTPE Profiling

The MAPLE-B maintains the following counters which sum the following FTPE parameters:

- MFTPP (MAPLE-B FFT Total Performance Parameter)—This parameter sums the total FFT samples of the FFTPE jobs. Once enabled, after every FFT job completion the FFTPE accumulates the recently completed job transform length with the current value, that is, MFTPP = MFTPP + Transform Length. If the job ends with an error the BD[TL_ID] is not added to the counter value.
- MFBDCP (MAPLE-B FFTPE BDs Counter Parameter)—This parameter sums the total number of FFTPE BDs completed by the MAPLE-B.
- MDTPP (MAPLE-B DFT Total Performance Parameter)—This parameter sums the total DFT samples of the DFTPE jobs. Once enabled, after every DFT job completion the DFTPE accumulates the recently completed job transform length with the current value, that is, MDTPP = MDTPP + Transform Length. If the job ends with an error the BD[TL_ID] will not be added to the counter value. If the DFTPE processes FFT jobs (see **Section 26.3.2.2.5**, *Using the DFTPE as FFTPE*), then the parameter will sum the Transform Lengths values of the FFT jobs it executes.
- MDBDCP (MAPLE-B DFTPE BDs Counter Parameter)—This parameter sums the total number of DFTPE BDs completed by the MAPLE-B.

Enabling and disabling these counters is done using the profiling toggle bit ([PRF_TGL]) in the FTPE BDs. The first time this bit is asserted, the MAPLE-B resets the FTPE parameter counters according to the MPEP enable bits (see **Section 26.4.2.3.9**, *MAPLE-B Profiling Enable Parameter (MPEP)*), and enables the count of the enabled parameters. On the next assertion of this bit the MAPLE-B disables the count, thus allowing the host to read the values of the counters. On the following assertion the MAPLE-B resets the counters and enables the count according to the MPEP enable bits and so on.

The host must initialize (or not) the DFTPE/FFTPE counters parameters. The MAPLE-B (once enabled) performs only the following for the enabled DFTPE/FFTPE parameters:

- Read relevant counter value.
- Add the required value (according to counter type).
- Write back the counter value.

Note: The BD which enables the count is counted in the profiling parameters while the BD which disables the count is not.

Note: Saturation event is not considered an error and therefore is accumulated as a valid BD.

Note: The [PRF_TGL] bit of the FFTPE BDs can enable/disable the FFTPE profiling parameters only, i.e. MFTPP and MFBDCP only.

Note: The [PRF_TGL] bit of the DFTPE BDs can enable/disable the DFTPE profiling parameters only, i.e. MDTPP and MDBDCP only.

26.3.2.3 CRC Operation

The MAPLE-B supports CRC checks and CRC calculations, using any of the four internally implemented fixed CRC polynomials, for block sizes of up to 16 KByte (128 Kbits). These operations are executed using the CRC Processing Element.

Note: The CRC processing executed by the CRC Processing Elements is not standard-related, and, therefore, is not affected by the MAPLE-B operation mode.

26.3.2.3.1 CRC Buffer-Descriptor Structure

A description of the CRC Buffer Descriptor structure expected by the host for the CRCPE BD rings is outlined in **Figure 26-35**. Some of the fields are status fields which are written by the MAPLE-B upon job completion. Other fields may be relevant only for certain configurations and will be ignored by the MAPLE-B if another configuration is used. The total length of the BD is 16 bytes, and its size does not depend on the configuration, that is, the next BD location should be located 16 bytes after the current BD address regardless of the BD configuration.

Offset BD_BASE + 0x0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OWNER	WRAP	DC	INT_EN						CHECK	UPDATE	RVRS_IN	RVRS_OUT	INV_OUT	CRC_POLY	
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			CRC_BS[													
W																

Offset	BD_BASE + 0x4															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CRC_IB															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CRC_IB															
W																

Offset	BD_BASE + 0x8															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CRC_INIT															
W																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CRC_INIT								TASKID							
W																

Figure 26-35. CRCPE Buffer Descriptor Structure

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

Figure 26-34. CRCPE Buffer Descriptor Structure (Continued)

Table 26-60. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0

Name	Description	
OWNER 31	**Owner Bit** When set by the host, MAPLE-B is the BD owner and the job is not complete. When cleared by MAPLE-B, the job is done and the host is the BD owner.	0 The host is BD owner 1 MAPLE-B is BD owner
WRAP 30	**Wrap Bit** When set by the host, MAPLE-B updates M<pe>BR(H/L)PAxP[BDR_RD_PTR[13:5]] with M<pe>BR(H/L)PAxP[BDR_BA[13:8]], that is, it expects to find the next BD at the Base Address of the ring. When cleared by the host, MAPLE-B expects to find the next BD at: M<pe>BR(H/L)PAxP[BDR_RD_PTR[13:5]] + BD size. <pe> stands for FF or DF	0 Next BD is at pointer + BD size. 1 Wrap to Base Address for next BD.
DC 29	**Data Coherency** Indicates whether the MAPLE-B should maintain data coherency when clearing the OWNER bit. If set the MAPLE-B assures Data transfer completion before clearing the OWNER bit, else it clears OWNER bit and shifts to next BD regardless of Data transfer status. For details, see **Section 26.3.1.3, *BDs Data Coherency Bit***.	0 No Data coherency is required. 1 Data coherency is required. The MAPLE-B assures Data transfer completion before clearing the OWNER bit and shifting to the next BD.
INT_EN 28	**Interrupt Enable** If set, MAPLE-B issues an interrupt/doorbell interrupt at the end of this job. If cleared no interrupt is issued.	0 End of job interrupt is NOT issued. 1 End of job interrupt is issued.
— 27–23	Reserved	
CHECK 22	**CRC Check** If set, MAPLE-B performs CRC check on the input buffer. A CRC pass/fail indication is described in the FAIL status bit. If reset, MAPLE-B performs CRC calculation on the input buffer. The CRC result is placed in the CRC_RSLT status field and (optional) at the end of the input buffer in the system memory.	0 MAPLE-B performs CRC calculation on the input buffer 1 MAPLE-B performs CRC check on the input buffer.
UPDATE 21	**Update Result in System Memory** Valid only if CHECK bit is reset. If set, MAPLE-B copies the CRC calculation result described in CRC_RSLT status field into the system memory at the end of the input buffer.	0 MAPLE-B does not update the input buffer with the CRC result. 1 MAPLE-B copies the CRC result described in the CRC_RSLT field into the system memory at the and of the input buffer
RVRS_IN 20	**Reverse Input CRC** If set, MAPLE-B performs byte reverse CRC calculation/check on the input data. For more details refer to **Section 26.3.2.3.4.1, *Byte Reverse CRC Processing***.	0 MAPLE-B performs straight forward CRC calculation/check. 1 MAPLE-B performs byte reverse CRC calculation/check.

Table 26-60. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x0 (Continued)

Name	Description	
RVRS_OUT 20	**Reverse Output** If set, MAPLE-B performs reveres operation on the CRC result. For more details refer to **Section 26.3.2.3.5.1**, *Reverse Output Operation*.	0 MAPLE-B outputs the CRC result as is. 1 MAPLE-B performs reverse on the CRC result.
INV_OUT 19	**Inverse Output** If set, MAPLE-B performs inverse operation on the CRC result. For more details refer to **Section 26.3.2.3.5.2**, *Inverse Output Operation*.	0 MAPLE-B does not perform inverse operation on the CRC result. 1 MAPLE-B performs inverse operation on the CRC result.
CRC_POLY[1:0] 17–16	**CRC Polynomial** Describes which Polynomial to use for the CRC calculation/check.	00 CRC24 with the following Polynomial: $D^{24} + D^{23} + D^6 + D^5 + D + 1$ 01 CRC24 with the following Polynomial: $D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$ 10 CRC16-CCITT with the following polynomial: $D^{16} + D^{12} + D^5 + 1$ 11 CRC16 with the following polynomial: $D^{16} + D^{15} + D^{14} + D^{11} + D^6 + D^5 + D^2 + D + 1$
— 15–14	Reserved	
CRC_BS[13:0] 13–0	**CRC Block Size** Describes the input buffer size (in bytes).	0 to 3FFF possible values (0 to 128Kbits)

Table 26-61. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x4

Name	Description
CRC_IB[31:3] 31–3	**CRC Input Buffer** This field points to the CRC input buffer location in system memory, where MAPLE-B is to read in order to perform CRC check/calculation. The address must be aligned to an 8 byte address.
— 2–0	Reserved

Table 26-62. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0x8

Name	Description
CRC_INIT[23:0] 31–8	**CRC Initialization Value** MAPLE-B uses this filed as the CRC initialization value for the CRC processing. In case the CRC polynomial is of size of 16 bits (CRC_POLY =(10) or (11)), then the valid bits for this filed are [15:0] only. for more details refer to section **Section 26.3.2.3.4.2**, *CRC Init value*
TASKID[7:0] 7–0	**Task ID Tag** Supplied by host and used by MAPLE-B if Serial RapidIO doorbell interrupt is required. For more details refer to **Section 26.3.3.3, *External Masters Support Using Serial RapidIO Doorbell***

Table 26-63. CRCPE Buffer Descriptor Fields Description of 4 Bytes at Offset 0xC

Name	Description	
FAIL 31	**CRC Fail** Valid only if [CHECK] bit is set and after [OWNER] bit is cleared by MAPLE-B. Status bit which indicates whether the CRC check has passed or failed.	0 CRC check has passed. 1 CRC check has failed.
— 30–24	Reserved	
CRC_RSLT[23:0] 23–0	**CRC Result** Valid only if [CHECK] bit is reset and after [OWNER] bit is cleared by MAPLE-B. Describes the result of the CRC calculation performed on the input buffer.	

26.3.2.3.2 Buffer Descriptors Notes

- Once the [OWNER] bit is set by the host, the BD content must not be changed. It can be re-written only after the [OWNER] bit was cleared by MAPLE-B.
- The CRC Input Buffer ([CRC_IB]) field in the CRC BD must be aligned to 8 bytes.
- All the status fields in the BD are updated by the MAPLE-B once the job is complete, that is, they are valid only after the owner bit is cleared by the MAPLE-B.
- When a host is writing a new BD to its ring, it must be done “bottom-up”, that is, the [OWNER] bit must be the last write access to the PSIF DRAM. If several BDs are written to the same ring, only the OWNER bit of the first written BD must be the last to be written.

26.3.2.3.3 CRC Input Data Structure

After parsing the CRC Buffer Descriptor, MAPLE-B starts fetching the input buffer into its internal memories for the CRC calculations/checks. The address of the input buffer should be specified in the [CRC_IB] field of the CRC BD. The CRC input buffer must be of size which is aligned to 8 bits, and its start address must be 8 bytes aligned. The CRC input buffer size must be specified in the [CRC_BS] field of the CRC BD. **Figure 26-35** shows an example of a 1224 bits block and how it should be placed in the system memory to be input for the MAPLE-B CRC processing:

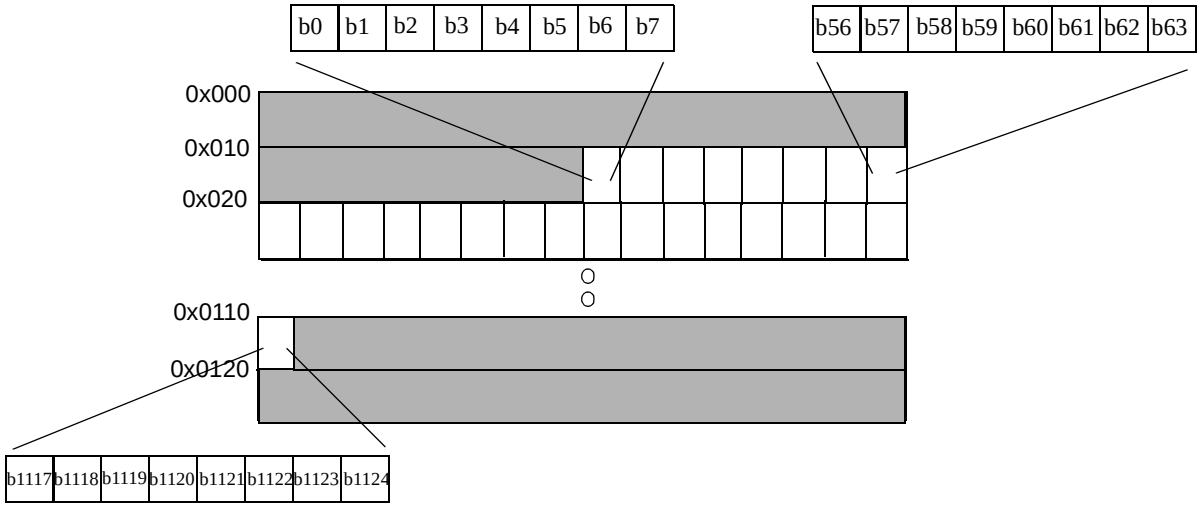


Figure 26-35. Example of Placing Block of 1224 Bits in the System Memory for CRC Calculation

If CRC calculations are required to execute on blocks that are not 8-bit aligned, zero padding bits must be added at the beginning of the input buffer before the first data bit. **Figure 26-36** shows an example of a 1220-bit block and how it should be placed in the system memory to be input for the MAPLE-B CRC processing:

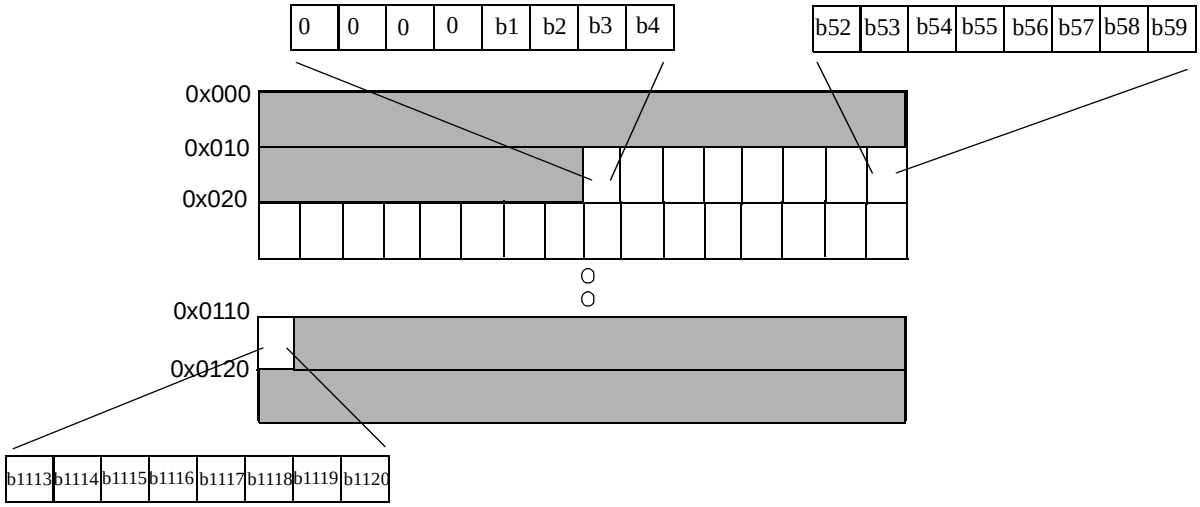


Figure 26-36. Example of Placing Block of 1220 Bits in the System Memory for CRC Calculation

26.3.2.3.4 CRC Processing

The MAPLE-B is capable of CRC processing of two types:

- CRC calculations. If the [CHECK] bit in the CRC BD is reset, the MAPLE-B performs CRC calculation on the input buffer and place the CRC calculation result in the [CRC_RSLT] field of the CRC BD.
- CRC checks. If the [CHECK] bit in the CRC BD is set, the MAPLE-B performs CRC check on the input buffer and updates the [FAIL] status bit in the CRC BD with the CRC check result. If the [FAIL] bit is set by the MAPLE-B, the CRC check failed. If the [FAIL] bit reset, the CRC check passed.

26.3.2.3.4.1 Byte Reverse CRC Processing

The CRCPE is capable of performing byte reverse CRC processing for both CRC calculations and CRC checks. By setting the [RVRS_IN] bit in the CRC BD, the MAPLE-B performs byte reverse CRC processing on the input data. **Figure 26-37** describe the CRC processing of a 64 bits vector with respect to the [RVRS_IN] bit configuration:

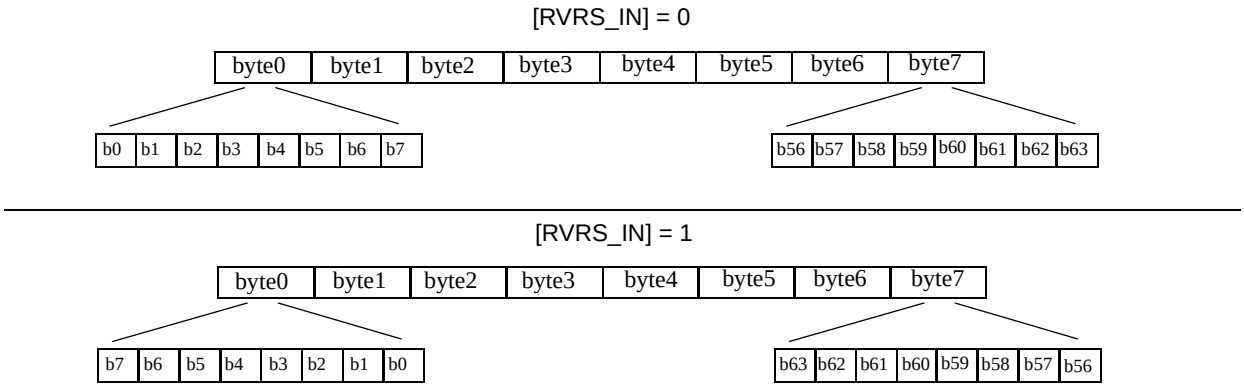


Figure 26-37. CRC Processing Order with Respect to [RVRS_IN] Bit

26.3.2.3.4.2 CRC Init value

The CRCPE is capable of initializing its CRC initial value with any configurable value. By configuring the [CRC_INIT] field of the CRCPE BD, MAPLE-B will copy the value into the CRCPE machine before the beginning of the CRC processing.

The configured value of the CRC_INIT filed must be with the same size as the CRC polynomial, i.e. if the configured polynomial is CRC16 or CRC16-CCITT ([CRC_POLY] = 10 or 11 respectively), the valid bit which will be copied by MAPLE-B into the CRCPE will be CRC_INIT[15:0] only. In this case bits [23:16] of the CRC_INIT files remains not valid. If the configured polynomial is any of the CRC24 polynomials ([CRC_POLY] = 00 or 01 respectively), then the whole CRC_INIT[23:0] field is valid.

26.3.2.3.4.3 CRC Polynomials

The CRCPE supports four types of CRC polynomials. The control of the CRC processing polynomial is done using the [CRC_POLY] field of the CRC BD according to the following:

if [CRC_POLY] = 00, the CRCPE performs CRC24 with the following polynomial:
$D^{24} + D^{23} + D^6 + D^5 + D + 1$

if [CRC_POLY] = 01, the CRCPE performs CRC24 with the following polynomial:
$D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1$

if [CRC_POLY] = 10, the CRCPE performs CRC16-CCITT with the following polynomial:
$D^{16} + D^{12} + D^5 + 1$

if [CRC_POLY] = 11, the CRCPE performs CRC16 with the following polynomial:
$D^{16} + D^{15} + D^{14} + D^{11} + D^6 + D^5 + D^2 + D + 1$

26.3.2.3.5 CRC Processing Results

The MAPLE-B is capable of performing the following processing of the CRC results:

26.3.2.3.5.1 Reverse Output Operation

When setting the [RVRS_OUT] bit of the CRCPE BD, the MAPLE-B performs reverse operation on the CRC result. The reverse operation means reflecting the result such as the Most Significant Bit (MSB) becomes the Least Significant Bit (LSB) and vice versa. The following figure describe the output reverse operation on a 16 bits CRC result:

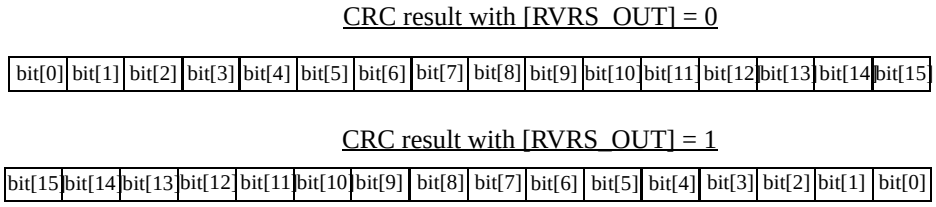


Figure 26-38. Reverse Output operation description

26.3.2.3.5.2 Inverse Output Operation

When setting the [INV_OUT] bit of the CRCPE BD, the MAPLE-B performs bit wise inverse operation on the CRC result. The following figure describe the output Inverse operation on a 16 bits CRC result:

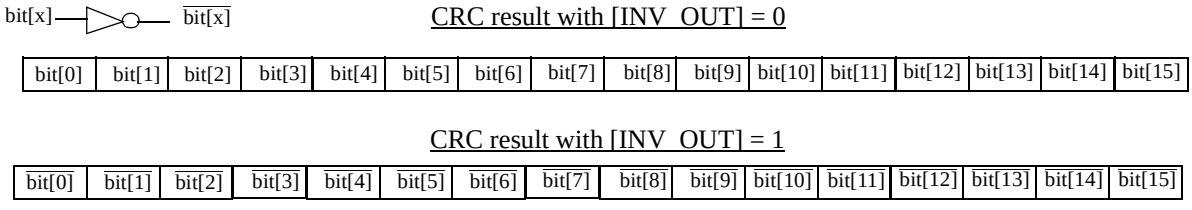


Figure 26-39. Inverse Output operation description

26.3.2.3.5.3 CRC Result Check/Calculate

If CRC check is required ([CHECK] = 1), MAPLE-B returns the CRC check PASS/FAIL indication using the [FAIL] status field of the CRC BD. If, after the [OWNER] bit is cleared, the [FAIL] status bit indication is set, the CRC check has failed. If it is reset, the CRC check has passed.

If CRC calculation is required, MAPLE-B returns the CRC calculation result in the [CRC_RSLT] field of the CRC BD. By setting the [UPDATE] bit in the CRC BD, MAPLE-B also updates the CRC result at the end of the input buffer in the system memory. **Figure 26-40** describes an example of input buffer after the CRC BD completion when the [UPDATE] bit is set.

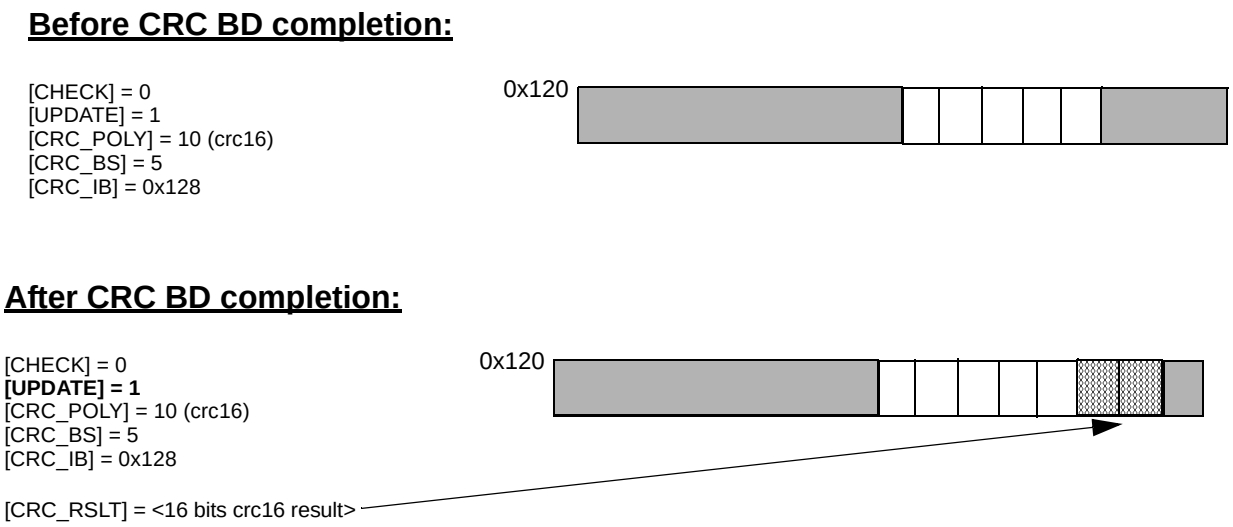


Figure 26-40. System Memory Update with CRC Result When [UPDATE] Bit is Set

26.3.3 :System Interface

The system interface supports ECC for memory, an interrupt system, and external masters.

26.3.3.1 Error Correction Code (ECC) Memory Support

The MAPLE-B includes the following internal memories that support ECC:

- PSIF DRAM. Data RAM memory which includes the MAPLE-B parameters, Buffer Descriptors and other internal configurations
- PSIF IRAM. Instruction RAM which includes code used by the internal programmable controller.
- FFTPE/DFTPE internal buffers.

Enabling/Disabling the ECC support is done only during the MAPLE-B initialization (using the API initialization routine), and during the reset routine of the MAPLE-B API. If ECC is enabled in the API, all memories supporting ECC are initialized by the MAPLE-B API (Application

ProgRAM Interface) and ECC check logic is enabled. The MAPLE-B ECC capabilities include code/data correction in case of one error and interrupt assertion in case of more than one error. If an ECC interrupt is asserted by the MAPLE-B, a reset routine must be initiated (using the MAPLE-B API).

26.3.3.2 Interrupts

The MAPLE-B supports up to 17 maskable interrupts, and 2 non-maskable interrupts. Detailed information on each type of interrupt is in the following subsections.

26.3.3.2.1 BD Rings Done Indication Interrupts

The MAPLE-B supports up to 16 BD ring regular interrupts. Each of the 8 High Priority BD rings and 8 Low Priority rings per PE can be mapped to any of the 16 interrupts using the M<*pe*>BR(H/L)PBxP[INT_TRGT] fields (<*pe*> = TV or FF or DF or CRC). Mapping a given ring to a given interrupt means an interrupt assertion on completion of any BD tasks related to that ring, assuming:

1. The [INT_EN] bit in the BD was set.
2. The mask bit related to that interrupt in PSPICMR[SFT_M] is set. (**Section 26.4.3.4**, *PSIF PIC Mask Register(PSPICMR)*)

Each of these 16 interrupt lines can be configured to be either an edge interrupt or a level interrupt using PSPICELR[SFT_EL] (**Section 26.4.3.3**, *PSIF PIC Edge/Level Register(PSPICELR)*). If edge configuration is chosen, the PSPICIACR[IAC] field determines the length of the edge interrupt (2, 4, 6, or 8 MAPLE-B clocks). If level configuration is chosen, the PSIF_IPIC holds an event for each of these interrupts in the PSIF PIC Event Register (PSPICER) (see **Section 26.4.3.2**, *PSIF PIC Event Register (PSPICER)* for details), which is set once an interrupt event has occurred. This bit is cleared by the host when accessing bits with a value of 1.

Each of these interrupts can be masked using a mask bit in the PSIF PIC Mask Register (PSPICMR) (**Section 26.4.3.4**, *PSIF PIC Mask Register(PSPICMR)*). Masking an interrupt means that only the interrupt line is not asserted; the event bit related to that interrupt is still set.

26.3.3.2.2 General Error Event Interrupt

The MAPLE-B supports a single interrupt line indicating one or more of the following events has occurred:

- Serial RapidIO doorbell interrupt event—Doorbell interrupt generated by MAPLE-B towards serial RapidIO controller completed with error. The doorbell interrupt did not reach its destination. The reason for the error can be read from the *ODSR* of the doorbell controller. This event can occur only if enabled during MAPLE-B initialization by setting the relevant bit in the mode parameter. For details, see **Section 26.3.3.3**.
- TVPE Configuration Error—The TVPE BD fields included the wrong configuration of either the BS field or the IN_D_STRUCT field. For details, see **Section 26.3.1.4.2**, *TVPE Buffer-Descriptor Structure*.
- DFTPE Configuration Error—The DFTPE BD fields included an un-supported value in its [TL_ID] field.
- FFTPE Configuration Error—The FFTPE BD fields included an un-supported value in its [TL_ID] field.

The interrupt line is triggered due to the assertion of one or more of the above events. For each of these events, a dedicated event bit is located in the PSPICER register. Reset of each of these bits is done when the host writing 1 to the event bit.

Each of the event bits related to this interrupt has a dedicated mask bit in the PSPICMR register that allows the host to control the events generating the general error interrupt.

The general error interrupt is level interrupt only, allowing the host to access the PSPICER in order to identify the source of the interrupt.

In the event that the general error interrupt indicates a BD configuration error, the host must read the status fields of all recently completed BDs in MAPLE-B to detect which BD generated the error.

26.3.3.2.3 ECC Error Interrupts

The MAPLE-B includes the following non-maskable error interrupts:

- Two ECC DED interrupts from the PSIF IRAM and DRAM memory modules.
- Two ECC DED interrupts from the FFTPE and DFTPE

Upon assertion of an ECC DED interrupt, the host must assume the MAPLE-B internal memory is corrupted, and hence must assert a soft reset to the MAPLE-B as described in the API User’s Manual.

26.3.3.3 External Masters Support Using Serial RapidIO Doorbell

The MAPLE-B can work with masters external to its DSP device using the Serial RapidIO interface. In order for the MAPLE-B to work with the Serial RapidIO interface and to enable the MAPLE-B receiving interrupt form RapidIO Doorbell controller, the relevant bit in the mode parameter of the MAPLE-B API must be set.

26.3.3.3.1 Serial RapidIO Doorbell Parameters Configuration

If an external master that is connected via the Serial RapidIO port wishes to master the MAPLE-B, the following parameters must be initialized:

- **M<pe>[1]BR(H/L)PBxP[HOST_ID]**—16 bit parameter Host ID to be used by MAPLE-B for the *ODDPR*[2] *[EDTGTROUTE:DTGTROUTE]* register in the Serial RapidIO Outbound Doorbell 0 Destination Port Register.
- **M<pe>BR(H/L)PBxP[P_TR_IF]**—4 bit parameter for the Port Target Interface in the RapidIO controller. This field is used by the MAPLE-B for the *ODDATR[TGINT]* register in the Outbound Doorbell 0 Destination Attributes Register.
- **M<pe>BR(H/L)PBxP[EXT_MST]**—2 bit parameter which describes whether the MAPLE-B current ring's master is connected directly to one of the MAPLE-B regular interrupt lines or the ring's master is external and connected to the SoC via the Serial RapidIO port.
- **SORDP0BAP[BA]**—32 bit address which describes the base address of the Doorbell controller Outbound registers.
- **HSP0BAP[BA]**—32 bit address which describes the address of the Hardware Semaphore used when trying to accesses the Serial RapidIO Doorbell controller. It is valid only if HSP0BAP[HS_EN] is set.
- **HSP0BAP[HS_EN]**—1 bit parameter which describes for the MAPLE-B whether it should use the Hardware Semaphore register when accessing the Doorbell controller. The bit should be set only if there are other possible masters on the Doorbell controller beside the MAPLE-B. This feature is added in order to avoid any coherency problems due to multiple masters accessing a Doorbell controller simultaneously.
- **MDHSIDCP[TV_HS_ID]**—8 bit parameter of TVPE External Hardware Semaphore ID, used by the MAPLE-B when accessing the Hardware Semaphore register for TVPE doorbell generation.
- **MDHSIDCP[DF_HS_ID]**—8 bit parameter of DFTPE External Hardware Semaphore ID, used by the MAPLE-B when accessing the Hardware Semaphore register for DFTPE doorbell generation.

1. <pe> stands for TV, FF or DF.
2. For mode details on the Doorbell controller registers see **Section 16.4.2**, *Doorbell Controller,* on page 16-109

- **MDHSIDCP[FF_HS_ID]**—8 bit parameter of FFTPE External Hardware Semaphore ID, used by the MAPLE-B when accessing the Hardware Semaphore register for FFTPE doorbell generation.
- **MDGCP[BD_PR]**—2 bit parameter filed which is used by the MAPLE-B to set the doorbell transaction priority in *OM0DATR[DTFLOWLVL]*.
- **MDGCP[RET]**—8 bit parameter which is used by the MAPLE-B for the *ODRETCR[RET]* field in the Doorbell controller. The *RET* field determines the number of times the doorbell controller will attempt to transmit the doorbell.

If a master in device #n needs to master a BD ring in the MAPLE-B of device #(n+2), the values of the relevant parameters should be:

— *MBDR(H/L)PBxP[HOST_ID] = 0xaaa*. The host_id value of device #n
— *MBDR(H/L)PBxP[EXT_MST] = 0x1*. This parameter indicates MAPLE-B to work with the door-bell controller of its SoC.

If the MAPLE-B in device#(n+2) is not the only master on the Doorbell controller, then:

— *HSP0BAP[HS_EN] = 0b1*. This assures that whenever the MAPLE-B (or any other master in the DSP) wishes to access the Doorbell controller, it first must lock the Semaphore, thus avoiding any coherency problem.
— *SORDP0BAP[BA]* must be initialized with the address of the *ODMR* register in the Serial RapidIO Doorbell controller.
— *HSP0BSP[BA]* must be initialized with the address of the HS register to be used when trying to use the Doorbell controller.
— *MDGCP[RET] = 0x5*. User definition. Describes the number of times the Doorbell controller tries to re-send the Doorbell before asserting its interrupt.
— *MDGCP[TV_HS_ID] = 0xA0*. User definition. Describe the MAPLE-B ID when accessing the Semaphore register for TVPE rings.
— *MDGCP[DF_HS_ID] = 0xA1*. User definition. Describe the MAPLE-B ID when accessing the Semaphore register for DFTPE rings.
— *MDGCP[FF_HS_ID] = 0xA2*. User definition. Describe the MAPLE-B ID when accessing the Semaphore register for FFTPE rings.

Note: If the *HSP0BAP[HS_EN] = 0b0*, that is, the MAPLE-B is the only doorbell controller master and therefore there is no need for external Semaphore, the MAPLE-B implements internal semaphore in order to avoid contention between its two internal RISC engines on the doorbell controller. This internal semaphore implementation requires no user configuration.

26.3.3.3.2 Serial RapidIO Configuration Information

Configuration of the two serial RapidIO controllers in the MSC8156E is done using a combination of registers in the General Configuration block, the HSSI, and the two RapidIO

controllers. Refer to **Chapter 8**, *General Configuration Registers*, **Chapter 15**, *High Speed Serial Interface (HSSI) Subsystem*, and **Chapter 16**, *Serial RapidIO Controller* for complete configuration details.

26.3.3.4 Operation Flow

Once the relevant parameters are configured to support an external host (BDR parameters, and Serial RapidIO parameters), the following flow should be maintained by the external host:

1. All input data related to the new jobs for the MAPLE-B should be placed in the MAPLE-B SoC memory map. The MAPLE-B can fetch its data only from its mapped system memory.
2. The new BDs should be placed in the external master BD ring in MAPLE-B internal memory.

Upon job completion, MAPLE-B outputs the relevant data; and if the [INT_EN] bit in the BD is set, the MAPLE-B initiates a Doorbell interrupt. The following flow describes the MAPLE-B actual accesses on the MBus interface if a Doorbell interrupt is to be issued and assuming the above parameters are configured:

1. If *HSP0BSP[HS_EN]* is set:
   - Write access to the External HS register in address *HSP0BSP[BA]* and the data is *MDGCP[<pe>_HS_ID]* according to the PE.
   - Read access from the External HS reg in address *HSP0BSP[BA]*. If the read data equals *MDGCP[<pe>_HS_ID]*, then Semaphore is locked for the PE, else MAPLE-B return to previous stage.
2. Read access to *ODSR* Doorbell controller register at address (*SORDP0BAP[BA]+0x4*). If bit 29 (DUB) is cleared, then the controller is not busy and the MAPLE-B can continue, else MAPLE-B continue polling this field until it is reset.
3. Write access to *ODSR* in order to clear the *ODSR[MER]*, *ODSR[RETE]*, *ODSR[PRT]* and *ODSR[EODI]* fields by writing 1 to them.
4. Write access to (*SORDP0BAP[BA]+0x18*), for the *ODDPR* register. The write data is {*MBDR(H/L)PBxP[HOST_ID]*,*0x0000*}.
5. Write access to (*SORDP0BAP[BA]+0x1C*), for the *ODDATR* register.The write data is {*0x01000*,*BD[CMP_RSN]*,*BD[TASK_ID]*}, where *BD[]* are fields from the relevant BD.This access enables the End-of-doorbell interrupt towards the MAPLE-B, defines the priority of the doorbell (second highest) and transfer the *CMP_RSN* and *TASK_ID* fields with the doorbell.
6. Write access to (*SORDP0BAP[BA]-+0x2C*), for the *ODRETCR* register. The write data is {*0x000000*,*MDGCP[RET]*}.

7. Write access to (*SORDP0BAP[BA]+0x0*), for the *ODMR* register. The write data is 0x1. This write access should initiate the Doorbell controller.

On doorbell operation completion, the doorbell controller generates an interrupt towards the MAPLE-B, which in turn, access the *ODSR* status register for completion status. The possible doorbell completion reasons are:

- Done response received. Operation completed with no errors.
- Error response received. Operation completed with error.
- Packet response time-out occurred. Operation completed with error.
- Retry limit was exceeded. Operation completed with error.

On doorbell completion interrupt assertion, the MAPLE-B performs the following:

1. Read access from the *ODSR* status register.
2. If *ODSR* status shows `done-response-received` and External HS is enabled (*HSP0BSP[HS_EN]* is set) then the MAPLE-B generates write access to External HS register with the *MDGCP[<pe>_HS_ID]* data in order to release the External HS.
3. If *ODSR* status shows other completion reason then `done-response-received`, the MAPLE-B generates its general error interrupt and if the External HS is enabled (*HSP0BSP[HS_EN]* is set) then the MAPLE-B generates write access to External HS register with the *MDGCP[<pe>_HS_ID]* data in order to release the External HS.

Note: Once Doorbell generation is completed, the MAPLE-B waits for the Doorbell interrupt in order to complete its flow and clear the OWNER bit; therefore the Doorbell interrupt must be enabled when MAPLE-B is working with external master.

Note: Generating Doorbell interrupts for every BD may cause a degradation in the MAPLE-B performance since it requires the MAPLE-B to configure the Doorbell controller. If the controller is busy, the MAPLE-B waits until it is free to receive its Doorbell transmission configuration. Therefore, it is recommended to enable such doorbell interrupts only once in several BDs.

Note: The Doorbell interrupt does not cover all possible physical Serial RapidIO errors. There are some additional errors indicated by a general Serial RapidIO interrupt line. This interrupt line is not connected to the MAPLE-B, thus cannot be detected by it.

26.3.3.5 MAPLE-B Internal Task Control

The MAPLE-B internal software programming is periodically rotatory, that is, after a BD completion it searches for the next BD; and if it fails to find a new BD, it hibernates for a time period as described in the MP_TPP parameter (**Section 26.4.2.1.3**, *MAPLE Timer Period Parameter (MP_TPP).*), and then starts a new search. For fast response of MAPLE-B to a new

BD, the MP_TPP[TMR_PRD] field should be set to minimum. There is no need for external intervention for the MAPLE-B to continuously process new BDs.

It is possible, however, to force the MAPLE-B to execute a certain routine. This is done using the PCR register (**Section 26.4.3.2**, *PSIF PIC Event Register (PSPICER)*). By asserting the PCR[FLG] bit with the relevant PCR[opcode] field, the internal PSIF RISC scheduler initiates a request to the RISC to execute one of the following routines.

- **MAPLE_parse_bd**. The MAPLE-B initiates the BD parse routines of all the PEs (TVPE,FFTPE,DFTPE,CRCPE), where each such routine scans the valid BD Rings of the relevant PE for a new valid BD to be execute.
- **MAPLE_parse_tvpe_bd**. The MAPLE-B initiates the TVPE BD parse routine, which scans the valid TVPE BD Rings for a new valid BD to execute.
- **MAPLE_parse_fftpe_bd**. The MAPLE-B initiates the FFTPE BD parse routine, which scans the valid FFTPE BD Rings for a new valid BD to execute.
- **MAPLE_parse_dftpe_bd**. The MAPLE-B initiates the DFTPE BD parse routine, which scans the valid DFTPE BD Rings for a new valid BD to execute.
- **MAPLE_parse_crcpe_bd**. The MAPLE-B initiates the CRCPE BD parse routine, which scans the valid CRCPE BD Rings for a new valid BD to execute.

Setting the FLG bit of the PGC is allowed only when the FLG bit is reset. A host accessing the PCR in order to initiate one of the above tasks must first make sure the FLG bit is reset. Accessing the PCR while FLG bit is set will results in ignoring the command. Such programming model of the PCR register requires to semaphore the PCR in case of multiple masters are allowed to access the PCR register.

26.3.3.6 Power Save

The MAPLE-B is designed for minimum power consumption while not in use. Each of the PEs in the MAPLE-B is designed to halt its internal clock toggling while in idle state, and to automatically re-toggle it when in use.

The PSIF is internally programed to hibernate whenever its PEs are busy, or when no new valid BDs are found. This is implemented using internal timers which are responsible for the periodic wake-up. After reset, the MAPLE-B internal RISCs enter into freeze mode until they are un-frozen by the MAPLE-B API.

26.3.3.7 Reset

The MAPLE-B has two types of reset:

26.3.3.7.1 External Hard/Soft Reset

On assertion of external reset, the MAPLE-B resets all its activities and state machines. If soft reset is asserted, some of its internal configuration registers keep their values for debugging purposes. After every external reset, the API must be initiated to initialize the MAPLE-B.

26.3.3.7.2 Internal Soft Reset

It is possible to initiate an internal soft reset to the MAPLE-B using only the soft reset routine in the API. When called, it will generate an internal soft reset to the whole MAPLE-B and initiate the MAPLE-B initialization sequence, which includes memories initialization and progRAM upload to the MAPLE-B internal memory according to the chosen operation mode.

26.3.4 Initialization Information

On deassertion of the power-on reset ($\overline{\text{PORESET}}$), the MAPLE-B is in freeze state, that is, the internal RISCs cannot access the internal instruction memory. At this stage, the MAPLE-B API must be initiated to activate the MAPLE-B.

The MAPLE-B API is primarily responsible for the following:

- Reset Data/Instruction internal memories if ECC is enabled, and enable the ECC feature. If no ECC is required, the API is to reset only the relevant parameters in the parameter RAM.
- Initialize the relevant parameters with default/boot values (as described in **Section 26.4.1.1**, *MAPLE-B Parameter RAM*)
- Upload the relevant progRAM into the MAPLE-B instruction RAM, according to the chosen operation mode.
- Unfreeze the operation of the MAPLE-B internal so that the MAPLE-B is ready to work.

The MAPLE-B API must be used after every external reset. It can also be used to generate a soft reset internally to the MAPLE-B.

26.4 Programming Model

The MAPLE-B includes the following functional memories:

- PSIF DRAM. Internal PSIF memory that includes the parameter RAM and the buffer descriptor rings ram
- TVPE Extrinsic memory. Internal memory for the TVPE extrinsic data. Can be accessed by the host for debugging.
- TVPE Blaster memory. Internal TVPE memory which includes the TVPE configuration and status registers
- FFTPE registers. Configuration and status registers
- DFTPE registers. Configuration and status registers

Each of these memories/registers are accessed by the MBus slave interface. The memory map description of the MBus slave and the SBus slave interfaces are outlined in detail in **Section 26.4.1.1**, *MAPLE-B Parameter RAM*.

The registers and parameters are treated in a similar fashion. The only difference is that the default values for the registers are the “Reset” values whereas the default values for parameters are the “Boot” values, and the default values for parameters apply only after the MAPLE-B initialization sequence is completed.

All registers/parameters are described as REG/PARAM_NAME<letter><offset>. The “letter” describes the number of times this register is implemented, and the offset field describes the offset value needed to be added to get the address of the next register. For example, MTVBRHPAxP (x = 0...7, offset 8) means that the following registers are implemented:

- MTVBRHPA0P at address 0x00000100
- MTVBRHPA1P at address 0x00000108
- ...
- MBDRHPA7P at address 0x00000138

Figure 26-41 describes the internal MAPLE-B MBus slave memory map partitioning.

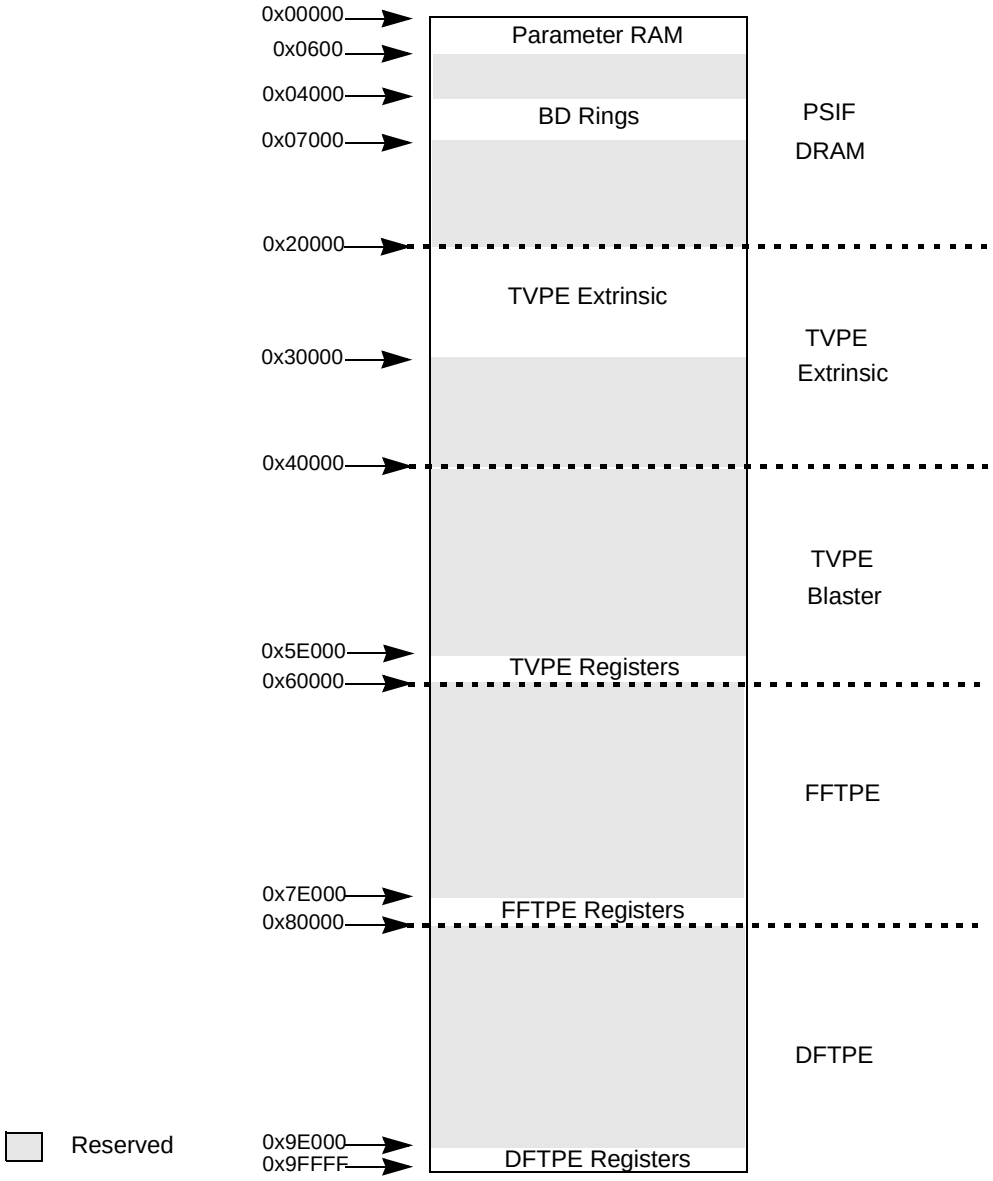


Figure 26-41. MAPLE-B MBus Slave Memory Map

26.4.1 Memory Map

The following sections described the location of the parameter RAM and PE registers.

26.4.1.1 MAPLE-B Parameter RAM

This section is a description of the MAPLE-B parameter RAM. **Table 26-64** lists the PSIF Parameter RAM contents.

Table 26-64. Parameter RAM Contents

Offset or Address	Parameters	Access	Boot Value[1]
General Parameters			
0x00000000	Reserved	N/A	N/A
0x00000004	MBDRCP. MAPLE BD Rings Configuration Parameter	R/W	0x00000000
0x00000010	MUCVP. MAPLE UCode Version Parameter	R	ucode version
0x00000018	MP_TPP. MAPLE Timer Period Parameter	R/W	Set by API
0x0000001C–0x0000007F	Reserved		
0x00000080–0x000000BF	MTVBRHPAxP. MAPLE TVPE BD Ring High Priority A <x> Parameter	R/W	0x00000000
	MTVBRHPBxP. MAPLE TVPE BD Ring High Priority B <x> Parameter	R/W	0x00000000
0x000000C0–0x000000FF	MTVBRLPAxP. MAPLE TVPE BD Ring Low Priority A <x> Parameter	R/W	0x00000000
	MTVBRLPBxP. MAPLE TVPE BD Ring Low Priority B <x> Parameter	R/W	0x00000000
0x00000100–0x0000013F	MFFBRHPAxP. MAPLE FFTPE BD Ring High Priority A <x> Parameter	R/W	0x00000000
	MFFBRHPBxP. MAPLE FFTPE BD Ring High Priority B <x> Parameter	R/W	0x00000000
0x00000140–0x0000017F	MFFBRLPAxP. MAPLE FFTPE BD Ring Low Priority A <x> Parameter	R/W	0x00000000
	MFFBRLPBxP. MAPLE FFTPE BD Ring Low Priority B <x> Parameter	R/W	0x00000000
0x00000180–0x000001BF	MDFBRHPAxP. MAPLE DFTPE BD Ring High Priority A <x> Parameter	R/W	0x00000000
	MDFBRHPBxP. MAPLE DFTPE BD Ring High Priority B <x> Parameter	R/W	0x00000000
0x000001C0–0x000001FF	MDFBRLPAxP. MAPLE DFTPE BD Ring Low Priority A <x> Parameter	R/W	0x00000000
	MDFBRLPBxP. MAPLE DFTPE BD Ring Low Priority B <x> Parameter	R/W	0x00000000
0x00000200–0x0000023F	MCRCBRHPAxP MAPLE CRCPE BD Ring High Priority A <x> Parameter	R/W	0x00000000
	MCRCBRHPBxP MAPLE CRCPE BD Ring High Priority B <x> Parameter	R/W	0x00000000
0x00000240–0x0000027F	MCRCBRLPAxP MAPLE CRCPE BD Ring Low Priority A <x> Parameter	R/W	0x00000000
	MCRCBRLHPBxP MAPLE CRCPE BD Ring Low Priority B <x> Parameter	R/W	0x00000000
TVPE Parameters			
0x00000300	MTSCCP. MAPLE Turbo Stop Criteria Configuration Parameter	R/W	0x00000000
0x00000304–0x0000037F	Reserved	—	—
0x00000380–0x000003C8	MTVPVxHCP. MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter	R/W	0x00000000
0x00000384–0x000003CC	MTVPVxLCP. MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter.	R/W	0x00000000
0x000003D0–0x000003D8	MTVPPCyP. MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter	R/W	0x00000000
0x000003E0–0x000003F0	MTVPVSxC0P. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 parameter	R/W	0x00000000
0x000003E4–0x000003F4	MTVPVSxC1P. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 parameter	R/W	0x00000000

Table 26-64. Parameter RAM Contents (Continued)

Offset or Address	Parameters	Access	Boot Value[1]
0x000003F8–0x000003FF	Reserved		
Profiling Parameters			
0x00000440	MTTPP. MAPLE-B Turbo Total Performance Parameter	R/W	0x00000000
0x00000444	MVTPP. MAPLE-B Viterbi Total Performance Parameter	R/W	0x00000000
0x00000448	MTBP. MAPLE-B Total BLER Parameter	R/W	0x00000000
0x0000044C	MTTBDCP. MAPLE-B TVPE Turbo BDs Counter Parameter	R/W	0x00000000
0x00000450	MFTPP. MAPLE-B FFT Total Performance Parameter	R/W	0x00000000
0x00000454	MFBDCP. MAPLE-B FFTPE BDs Counter Parameter	R/W	0x00000000
0x00000458	MDTPP. MAPLE-B DFT Total Performance Parameter	R/W	0x00000000
0x0000045C	MDBDCP. MAPLE-B DFTPE BDs Counter Parameter	R/W	0x00000000
0x00000460	MPEP. MAPLE-B Profiling Enable Parameter	R/W	0x00000000
0x00000464–0x0000047F	Reserved		
Serial RapidIO Doorbell Support Attributes Parameters			
0x00000480	SORDP0BAP. Serial RapidIO Outbound RapidIO Doorbell Base Address Parameter	R/W	0x00000000
0x00000484	Reserved.		
0x00000488	HSP0BAP. Hardware Semaphore Base Address Parameter	R/W	0x00000000
0x0000048C	Reserved.		
0x00000490	MDHSIDCP. MAPLE-B Doorbell Hardware Semaphore ID Configuration Parameter	R/W	0x00000000
0x00000494	MDGCP. MAPLE-B Doorbell General Configuration Parameter	R/W	0x00000000
0x00000498–0x00003FFF	Reserved.	R/W	0x00000000
BD Rings			
0x00004000–0x00006FFF	BD Rings Memory	R/W	0x00000000
Reserved			
0x00007000-0x000077FF	Reserved.		
0x00007800-0x000079FF	Reserved for SmartDSP OS Driver.		
0x00007A00-0x0001FFFF	Reserved.		

1. All the parameters in the parameter RAM are initialized during the MAPLE-B API initialization sequence, and are not hardware reset values.

26.4.1.2 TVPE Registers

Table 26-65 lists the TVPE registers.

Table 26-65. TVPE Registers

Offset or Address	Register	Access	Reset Value
Configuration Registers			
0x0005E000	TVPEC0R. TVPE Configuration 0 Register	R/W	0x00000000
0x0005E008	TVSI0CR. TVPE Symbol Identification 0 Configuration Register	R/W	0x00000000
0x0005E00C	TVSI1CR. TVPE Symbol Identification 1 Configuration Register	R/W	0x00000000
0x0005E010–0x0005E023	TVTTSIxCR. TVPE Tail Symbol Identification X Configuration Register	R/W	0x00000000
0x0005E024–0x0005E037	Reserved		
0x0005E03C	TVAQCR. TVPE Aposteriori Quality Configuration Register	R/W	0x00000000
0x0005E040	TVVPVG0CR. TVPE Viterbi Polynomial Vector Generation 0 Configuration Register	R/W	0x00000000
0x0005E044	TVVPVG1CR - TVPE Viterbi Polynomial Vector Generation 1 Configuration Register	R/W	0x00000000
0x0005E140	Reserved		
Control Registers			
0x0005E8D4–0x0005E8DF	Reserved		
0x0005EA20–0x0005EA5F	Reserved		
0x0005EAA–0x0005EAFF	Reserved		
Status Registers			
0x0005ED00	TVPESR. TVPE Decoder Status Register	R	0x00000000

26.4.1.3 FFTPE Registers

Table 26-66 lists the FFTPE registers.

Table 26-66. FFTPE Registers

Offset or Address	Register	Access	Reset Value
0x0007E140–0x0007E148	FFTPEDSRx. FFTPE Data Size Registers	R/W	0x00000000
0x0007E238	FFTPESTR. FFTPE Status Register	R	0x0000_0000
0x0007E23C	FFTPESCLSTR. FFTPE Scaling Status Register	R	0x0000_0000
0x0007E240	FFTPESSTR0. FFTPE Saturation Status Register 0	R	0x0000_0000
0x0007E244	FFTPESSTR1. FFTPE Saturation Status Register 1	R	0x0000_0000
0x0007E248	FFTPESSTR2. FFTPE Saturation Status Register 2	R	0x0000_0000
0x0007E24C	FFTPESSTR3. FFTPE Saturation Status Register 3	R	0x0000_0000

26.4.1.4 DFTPE Registers

Table 26-67 lists the DFTPE registers.

Table 26-67. DFTPE Registers

Offset or Address	Register	Access	Reset Value
0x0009E140–0x0009E144	DFTPEDSRx. DFTPE Data Size Registers	R/W	0x00000000
0x0009E230	DFTPEIR. DFTPE Interrupt Register	R	0x0000_0000
0x0009E238	DFTPESTR. DFTPE Status Register	R	0x0000_0000
0x0009E23C	DFTPESCLSTR. DFTPE Scaling Status Register	R	0x0000_0000
0x0009E240	DFTPESSTR0. DFTPE Saturation Status Register 0	R	0x0000_0000
0x0009E244	DFTPESSTR1. DFTPE Saturation Status Register 1	R	0x0000_0000
0x0009E248	DFTPESSTR2. DFTPE Saturation Status Register 2	R	0x0000_0000
0x0009E24C	DFTPESSTR3. DFTPE Saturation Status Register 3	R	0x0000_0000

26.4.2 Detailed Descriptions

26.4.2.1 Buffer Descriptors (BD)

26.4.2.1.1 MAPLE BD Rings Configuration Parameter (MBDRCP)

Offset 0x00000004 (MBDRCP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		CRC_R_LOOP				CRC_HL_ARB				DF_R_LOOP				DF_HL_ARB		
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R		FF_R_LOOP				FF_HL_ARB				TV_R_LOOP				TV_HL_ARB		
W																
Boot	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

Figure 26-42. MAPLE BD Rings Configuration Parameter

This parameter determines the number of high/low BD rings used and the arbitration scheme between the high/low BD rings for each of the PEs.

Table 26-68. MAPLE BD Rings Configuration Parameter Fields Description

Field	Description	Settings
— 31	Reserved	
CRC_R_LOOP 30–28	**CRCPE Ring Loop** The MAPLE-B assumes the potential valid high/low rings of the CRCPE are located only between ring #0 and ring #(CRC_R_LOOP). For more details see **Section 26.3.1.2**, *BD Arbitration,* on page 26-8.	000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111 Ring #0 to ring #7 are potentially valid
— 27	Reserved	
CRC_HL_ARB 26–24	**CRCPE High Low Arbitration Sequence** Determines the order for the CRCPE to scan for its next job in the CRCPE High BD rings and Low BD rings.	000 MAPLE-B scans the High BD rings of the CRCPE only. 001 MAPLE-B scans for next valid BD of the CRCPE as follows: H-L-H-L... i.e., after executing one job from any of the High priority rings of the CRCPE, it scans for the next job in the Low priority rings of the CRCPE. 010 MAPLE-B scans for next valid BD of the CRCPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the CRCPE, it scans the next job in the Low priority rings of the CRCPE. 011 to 111 Reserved
— 23	Reserved	

Table 26-68. MAPLE BD Rings Configuration Parameter Fields Description (Continued)

Field	Description	Settings
DF_R_LOOP 22–20	**DFTPE Ring Loop** The MAPLE-B assumes the potential valid high/low rings of the DFTPE are located only between ring #0 and ring #(DF_R_LOOP). For details, see **Section 26.3.1.1**, *BD Rings Arbitration*, on page 26-6."	000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111 Ring #0 to ring #7 are potentially valid
— 19	Reserved	
DF_HL_ARB 18–16	**DFTPE High Low Arbitration Sequence** Determines the order for the DFTPE to scan for its next job in the DFTPE High BD rings and Low BD rings.	000 MAPLE-B scans the High BD rings of the DFTPE only. 001 MAPLE-B scans for next valid BD of the DFTPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the DFTPE, it scans for the next job in the Low priority rings of the DFTPE. 010 MAPLE-B scans for next valid BD of the DFTPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the DFTPE, it scans the next job in the Low priority rings of the DFTPE. 011 to 111 Reserved
— 15	Reserved	
FF_R_LOOP 14–12	**FFTPE Ring Loop** The MAPLE-B assumes the potential valid high/low rings of the FFTPE are located only between ring #0 and ring #(FF_R_LOOP). For more details see **Section 26.3.1.1**, *BD Rings Arbitration*, on page 26-6.	000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111 Ring #0 to ring #7 are potentially valid
— 11	Reserved	
FF_HL_ARB 10–8	**FTPE High Low Arbitration Sequence** Determines the order for the FFTPE to scan for its next job in the FFTPE High BD rings and Low BD rings.	000 MAPLE-B scans the High BD rings of the FFTPE only. 001 MAPLE-B scans for next valid BD of the FFTPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the FFTPE, it scans for the next job in the Low priority rings of the FFTPE. 010 MAPLE-B scans for next valid BD of the FFTPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the FFTPE, it scans the next job in the Low priority rings of the FFTPE. 011 to 111 Reserved
— 7	Reserved	
TV_R_LOOP 6–4	**TVPE Ring Loop** The MAPLE-B assumes the potential valid high/low rings of the TVPE are located only between ring #0 and ring #(TV_R_LOOP). For more details see **Section 26.3.1.1**, *BD Rings Arbitration*, on page 26-6.	000 Only ring #0 is potentially valid 001 Ring #0 to ring #1 are potentially valid ... 111 Ring #0 to ring #7 are potentially valid
— 3	Reserved	

Table 26-68. MAPLE BD Rings Configuration Parameter Fields Description (Continued)

Field	Description	Settings
TV_HL_ARB 2–0	**TVPE High Low Arbitration Sequence** Determines the order for the TVPE to scan for its next job in the TVPE High BD rings and Low BD rings.	000 MAPLE-B scans the High BD rings of the TVPE only. 001 MAPLE-B scans for next valid BD of the TVPE as follows: H-L-H-L... that is, after executing one job from any of the High priority rings of the TVPE, it scans for the next job in the Low priority rings of the TVPE. 010 MAPLE-B scans for next valid BD of the TVPE as follows: H-H-L-H-H-L... that is, after executing two jobs from any of the High priority rings of the TVPE, it scans the next job in the Low priority rings of the TVPE. 011 to 111 Reserved

26.4.2.1.2 MAPLE UCode Version Parameter (MUCVP)

Offset 0x00000010 (MUCVP) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UCV[31:16]															
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UCV[15:0]															
W																
Initial	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 26-43. MAPLE UCode Version Parameter

This parameter includes the internal RISC engine ucode version. It is written during the MAPLE-B activation using its API.

Table 26-69. MAPLE UCode Version Parameter Fields Description

Name	Description
UCV 31–0	**Ucode Version** This field describes the ucode and API version used to initiate the MAPLE-B.

26.4.2.1.3 MAPLE Timer Period Parameter (MP_TPP).

Offset 0x00000018 (MP_TPP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											TMR_PRD					
W																
Initial	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 26-44. MAPLE Timer Period Parameter

This parameter sets the time periods between each time the internal RISC engines are being called in order to search for new Buffer Descriptor. For more details refer to **Section 26.3.3.5**, *MAPLE-B Internal Task Control*

Table 26-70. MAPLE Timer Period Parameter Fields Description

Name	Description
— 31–6	Reserved.
TMR_PRD 5–0	**Timer Period** Determines the time (MAPLE clock cycles) which cause the internal RISC engine to search for the next Buffer Descriptor. The number of MAPLE clock cycles is determined according to the following: <MAPLE clock cycles> = (1 + TMR_PRD) * 1024. For more details refer to **Section 26.3.3.5**, *MAPLE-B Internal Task Control*

26.4.2.1.4 MAPLE TVPE BD Ring High Priority A <x> Parameter (MTVBRHPAxP)

Offset 0x00000080 (MTVBRHPA0P) Access: User read/write
offset 8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:3]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-45. MAPLE TVPE BD Ring High Priority A <x> Parameter

These parameters, along with **MTVBRHBxP,** describe the attributes of a High-Priority TVPE BD ring. There are 8 such parameters, one for each possible High-Priority TVPE BD ring.

Table 26-71. MAPLE TVPE BD Ring High Priority A <x> Parameter Fields Description

Field	Description	Settings
— 31–30	Reserved	
BDR_BA[13:8] 29–24	**D Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes. Therefore the field range is 13:8. The B.A. should be within the 12 KBytes of BD memory in the DRAM. The given address is relative to the start address of the BD memory in the DRAM.	0x00 to 0x2F BD Ring Base Address for TVPE High Priority Ring x. (stands for base address 0x0000 to 0x2F00)
— 23–14	Reserved	
BDR_RD_PTR[13:3] 13–3	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-incriminated by the MAPLE-B after executing the BD. The host should use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field with the same value as BDR_BA, and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results. The pointer should be relative to the start address of the BD memory in the DRAM. Therefore, the range of addresses should be:	0x000 to 0x7FF Read Pointer for TVPE High Priority Ring x
— 2–0	Reserved	

26.4.2.1.5 MAPLE TVPE BD Ring High Priority B <x> Parameter (MTVBRHPBxP)

Offset 0x00000084(MTVBRHPB0P) Access: User read/write
offset 8
range x = 0...7

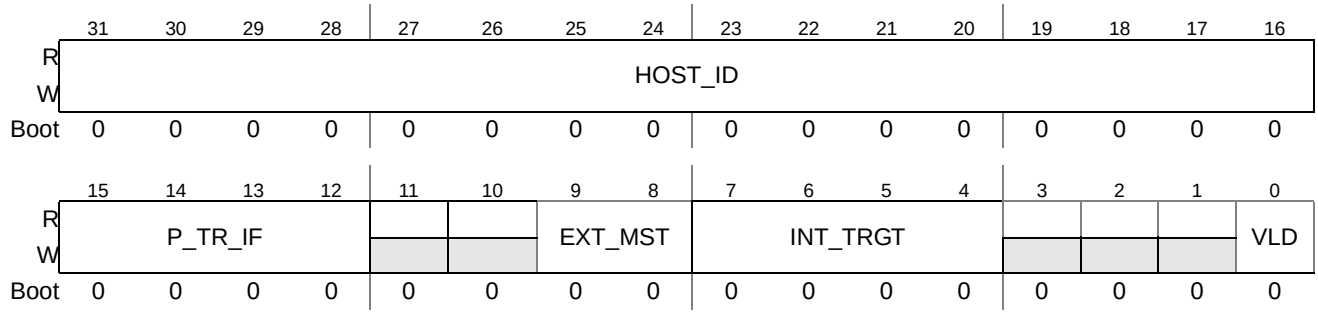


Figure 26-46. MAPLE TVPE BD Ring High Priority B <x> Parameter

These parameters, along with **MTVBRHAxP,** describe the attributes of a High-Priority TVPE BD rings. There are 8 such parameters, one for each possible High-Priority TVPE BD ring.

Table 26-72. MAPLE TVPE BD Ring High Priority A <x> Parameter Fields Description

Field	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an External (to SoC) master connected to the SoC by Serial RapidIO port and is the owner of this ring. This field is valid only if MTVBRHPBxP[EXT_MST] equals 0b01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	
EXT_MST 9–8	**External Master** Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and an interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and the Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*.	0b00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MTVBRHPBxP[INT_TRGT]. 0b01 The master of this ring is External (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated using MTVBRHPBxP[HOST_ID] (**Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*). 0b10 Reserved. 0b11 Reserved
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MTVBRHPBxP[EXT_MST] equals 0b00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	

Table 26-72. MAPLE TVPE BD Ring High Priority A <x> Parameter Fields Description

Field	Description	Settings
VLD 0	**Valid Bit** Describes if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.1.6 MAPLE TVPE BD Ring Low Priority A <x> Parameter (MTVBRLPAxP)

Offset 0x000000C0 (MTVBRLPA0P) Access: User read/write
offset 8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:3]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-47. MAPLE TVPE BD Ring Low Priority A <x> Parameter

These parameters, along with **MTVBRLBxP,** describe the attributes of a Low-Priority TVPE BD ring. There are 8 such parameters, one for each possible Low-Priority TVPE BD ring.

Table 26-73. MAPLE TVPE BD Ring Low Priority A <x> Parameter Fields Description

Field	Description	Settings
— 31–30	Reserved	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM. The given address is relative to the start address of the BD memory in the DRAM.	0x00 to 0x2F BD Ring Base Address for TVPE Low Priority Ring x. (stands for base address 0x0000 to 0x2F00)
— 23–14	Reserved	

Table 26-73. MAPLE TVPE BD Ring Low Priority A <x> Parameter Fields Description

Field	Description	Settings
BDR_RD_PTR[13:3] 13–3	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-incriminated by the MAPLE-B, after executing the BD. The host should use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA, and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results. The pointer should be relative to the start address of the BD memory in the DRAM, therefore the range of addresses should be:	0x000 to 0x7FF Read Pointer for TVPE Low Priority Ring x
— 2–0	Reserved	

26.4.2.1.7 MAPLE TVPE BD Ring Low Priority B <x> Parameter (MTVBRLPBxP)

Offset 0x000000C4(MTVBRLPB0P) Access: User read/write
offset 8
range x = 0...7

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	P_TR_IF						EXT_MST		INT_TRGT							VLD
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-48. MAPLE TVPE BD Ring Low Priority B <x> Parameter

These parameters, along with **MTVBRLAxP,** describe the attributes of a Low-Priority TVPE BD ring. There are 8 such parameters, one for each possible Low-Priority TVPE BD ring.

Table 26-74. MAPLE TVPE BD Ring Low Priority B <x> Parameter Fields Description

Field	Description	Settings
HOST_ID 31–16	**16 Bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by Serial RapidIO port and is the owner of this ring. This field is valid only if MTVBRLPBxP[EXT_MST] equals 0b01.	

Table 26-74. MAPLE TVPE BD Ring Low Priority B <x> Parameter Fields Description

Field	Description	Settings
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	
EXT_MST 9–8	**External Master** Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*.	0b00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MTVBRLPBxP[INT_TRGT]. 0b01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated using MTVBRLPBxP[HOST_ID] (**Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*). 0b10 Reserved. 0b11 Reserved
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MTVBRLPBxP[EXT_MST] equals 0b00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved.	
VLD 0	**Valid Bit** Describe if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore, any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.1.8 MAPLE FFTPE BD Ring High Priority A <x> Parameter (MFFBRHPAxP)

Offset 0x00000100 (MFFBRHPA0P) Access: User read/write
offset 8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:3]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-49. MAPLE FFTPE BD Ring High Priority A <x> Parameter

These parameters, along with **MFFBRHPBxP,** describe the attributes of a High-Priority FFTPE BD ring. There are 8 such parameters, one for each possible High-Priority FFTPE BD ring.

Table 26-75. MAPLE FFTPE BD Ring High Priority A <x> Parameter Fields Description

Field	Description	
— 31–30	Reserved	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM.	The given address is relative to the start address of the BD memory in the DRAM. 0x00 to 0x2F BD Ring Base Address for FFTPE High Priority Ring x. Stands for base address 0x0000 to 0x2F00.
— 23–14	Reserved	
BDR_RD_PTR[13:3] 13–3	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-incriminated by the MAPLE-B, after executing the BD. The host should use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA, and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results.	The pointer should be relative to the start address of the BD memory in the DRAM, therefore the range of addresses should be: 0x000 to 0x7FF Read Pointer for FFTPE High Priority Ring x
— 2–0	Reserved	

26.4.2.1.9 MAPLE FFTPE BD Ring High Priority B <x> Parameter (MFFBRHPBxP)

Offset 0x00000104(MFFBRHPB0P) Access: User read/write
offset 8
range x = 0...7

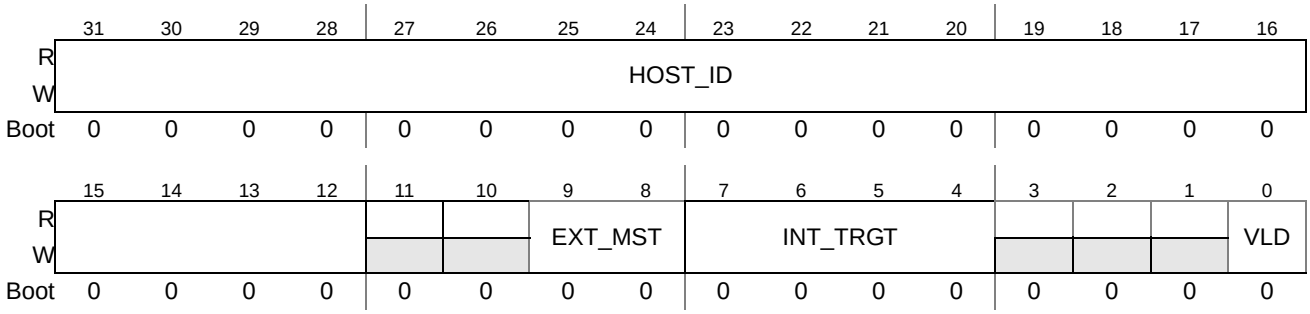


Figure 26-50. MAPLE FFTPE BD Ring High Priority B <x> Parameter

These Parameters, along with **MFFBRHAxP,** describes the attributes of a High-Priority FFTPE BD rings. There are 8 such Parameters, one for each possible High-Priority FFTPE BD ring.

Table 26-76. MAPLE FFTPE BD Ring High Priority B <x> Parameter Fields Description

Field	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by Serial RapidIO port and is the owner of this ring. This field is valid only if MFFBRHPBxP[EXT_MST] equals 0b01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	
EXT_MST 9–8	**External Master** Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*.	0b00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MFFBRHPBxP[INT_TRGT]. 0b01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated using MFFBRHPBxP[HOST_ID] (**Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*). 0b10 Reserved. 0b11 Reserved
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MFFBRHPBxP[EXT_MST] equals 0b00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	

Table 26-76. MAPLE FFTPE BD Ring High Priority B <x> Parameter Fields Description

Field	Description	Settings
VLD 0	**Valid Bit** Describe if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.1.10 MAPLE FFTPE BD Ring Low Priority A <x> Parameter (MFFBRLPAxP)

Offset 0x00000140 (MFFBRLPA0P) Access: User read/write
offset 8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:3]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-51. MAPLE FFTPE BD Ring Low Priority A <x> Parameter

These parameters, along with **MFFBRLBxP,** describe the attributes of a Low-Priority FFTPE BD ring. There are 8 such parameters, one for each possible Low-Priority FFTPE BD ring.

Table 26-77. MAPLE FFTPE BD Ring Low Priority A <x> Parameter Fields Description

Field	Description	Settings
— 31–30	Reserved	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM.	The given address is relative to the start address of the BD memory in the DRAM. 0x00 to 0x2F BD Ring Base Address for FFTPE Low Priority Ring x (stands for base address 0x0000 to 0x2F00)
— 23–14	Reserved	

Table 26-77. MAPLE FFTPE BD Ring Low Priority A <x> Parameter Fields Description

Field	Description	Settings
BDR_RD_PTR[13:3] 13–3	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-incriminated by the MAPLE-B, after executing the BD. The host should use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA, and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results.	The pointer should be relative to the start address of the BD memory in the DRAM, therefore, the range of addresses should be: 0x000 to 0x7FF Read Pointer for FFTPE Low Priority Ring x
— 2–0	Reserved	

26.4.2.1.11 MAPLE FFTPE BD Ring Low Priority B <x> Parameter (MFFBRLPBxP)

Offset 0x00000144(MFFBRLPB0P) Access: User read/write
offset 8
range x = 0...7

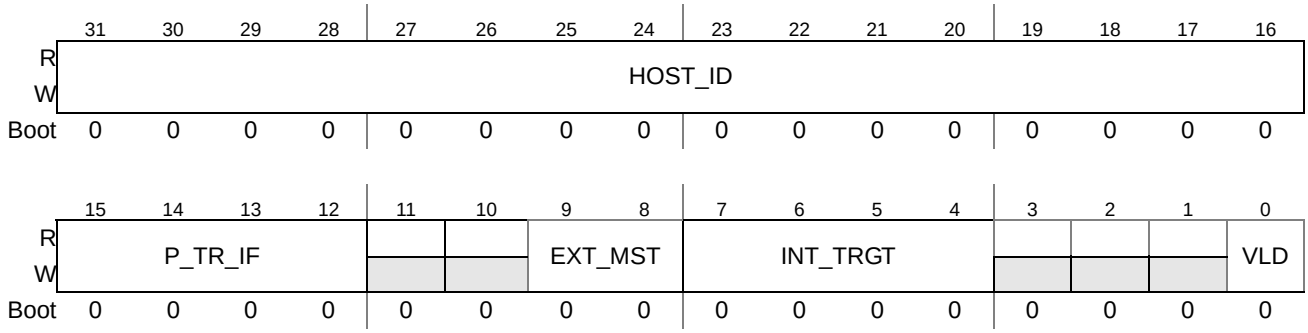


Figure 26-52. MAPLE FFTPE BD Ring Low Priority B <x> Parameter

These parameters, along with **MFFBRLAxP,** describe the attributes of a Low-Priority FFTPE BD ring. There are 8 such parameters, one for each possible Low-Priority FFTPE BD ring.

Table 26-78. MAPLE FFTPE BD Ring Low Priority B <x> Parameter Fields Description

Field	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by the Serial RapidIO port and owner of this ring. This field is valid only if MFFBRLPBxP[EXT_MST] equals 0b01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	

Table 26-78. MAPLE FFTPE BD Ring Low Priority B <x> Parameter Fields Description

Field	Description	Settings
EXT_MST 9–8	External Master. Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*.	0b00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MFFBRLPBxP[INT_TRGT]. 0b01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated using MFFBRLPBxP[HOST_ID](**Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*). 0b10 Reserved. 0b11 Reserved.
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MFFBRLPBxP[EXT_MST] equals 0b00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	
VLD 0	**Valid Bit** Describe if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.1.12 MAPLE DFTPE BD Ring High Priority A <x> Parameter (MDFBRHPAxP)

Offset 0x00000180 (MDFBRHPA0P) Access: User read/write
offset 8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:3]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-53. MAPLE DFTPE BD Ring High Priority A <x> Parameter

These parameters, along with **MDFBRHPBxP,** describe the attributes of a High-Priority DFTPE BD ring. There are 8 such parameters, one for each possible High-Priority DFTPE BD ring.

Table 26-79. MAPLE DFTPE BD Ring High Priority A <x> Parameter Fields Description

Field	Description	Settings
— 31–30	Reserved	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM.	The given address is relative to the start address of the BD memory in the DRAM.0x00 to 0x2F BD Ring Base Address for DFTPE High Priority Ring x. (stands for base address 0x0000 to 0x2F00
— 23–14	Reserved	
BDR_RD_PTR[13:3] 13–3	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-incriminated by the MAPLE-B, after executing the BD. The host should use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA, and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results.	The pointer should be relative to the start address of the BD memory in the DRAM, therefore the range of addresses should be: 0x000 to 0x7FF Read Pointer for DFTPE High Priority Ring x
— 2–0	Reserved	

26.4.2.1.13 MAPLE DFTPE BD Ring High Priority B <x> Parameter (MDFBRHPBxP)

Offset 0x00000184(MDFBRHPB0P) Access: User read/write
offset 8
range x = 0...7

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	P_TR_IF						EXT_MST		INT_TRGT							VLD
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-54. MAPLE DFTPE BD Ring High Priority B <x> Parameter

These parameters, along with **MDFBRHAxP,** describe the attributes of a High-Priority DFTPE BD ring. There are 8 such parameters, one for each possible High-Priority DFTPE BD ring.

Table 26-80. MAPLE DFTPE BD Ring High Priority B <x> Parameter Fields Description

Field	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by the Serial RapidIO port and owner of this ring. This field is valid only if MDFBRHPBxP[EXT_MST] equals 0b01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	
EXT_MST 9–8	**External Master** Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*.	0b00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MDFBRHPBxP[INT_TRGT]. 0b01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated using MDFBRHPBxP[HOST_ID] (**Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*). 0b10 Reserved. 0b11 Reserved
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MDFBRHPBxP[EXT_MST] equals 0b00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	

Table 26-80. MAPLE DFTPE BD Ring High Priority B <x> Parameter Fields Description

Field	Description	Settings
VLD 0	**Valid Bit** Describe if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.1.14 MAPLE DFTPE BD Ring Low Priority A <x> Parameter (MDFBRLPAxP)

Offset 0x000001C0 (MDFBRLPA0P) Access: User read/write
offset 8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:3]													
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-55. MAPLE DFTPE BD Ring Low Priority A <x> Parameter

These parameters, along with **MDFBRLBxP,** describe the attributes of a Low-Priority DFTPE BD ring. There are 8 such parameters, one for each possible Low-Priority DFTPE BD ring.

Table 26-81. MAPLE DFTPE BD Ring Low Priority A <x> Parameter Fields Description

Field	Description	Settings
— 31–30	Reserved.	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM.	The given address is relative to the start address of the BD memory in the DRAM. 0x00 to 0x2F—BD Ring Base Address for DFTPE Low Priority Ring x. Stands for base address 0x0000 to 0x2F00.
— 23–14	Reserved	

Table 26-81. MAPLE DFTPE BD Ring Low Priority A <x> Parameter Fields Description

Field	Description	Settings
BDR_RD_PTR[13:3] 13–3	—BD Ring Read Pointer. This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-incriminated by the MAPLE-B, after executing the BD. The host should use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA, and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results.	The pointer should be relative to the start address of the BD memory in the DRAM, therefore the range of addresses should be: 0x000 to 0x7FF—Read Pointer for DFTPE Low Priority Ring x
— 2–0	Reserved	

26.4.2.1.15 MAPLE DFTPE BD Ring Low Priority B <x> Parameter (MDFBRLPBxP)

Offset 0x000001C4(MDFBRLPB0P) Access: User read/write
offset 8
range x = 0...7

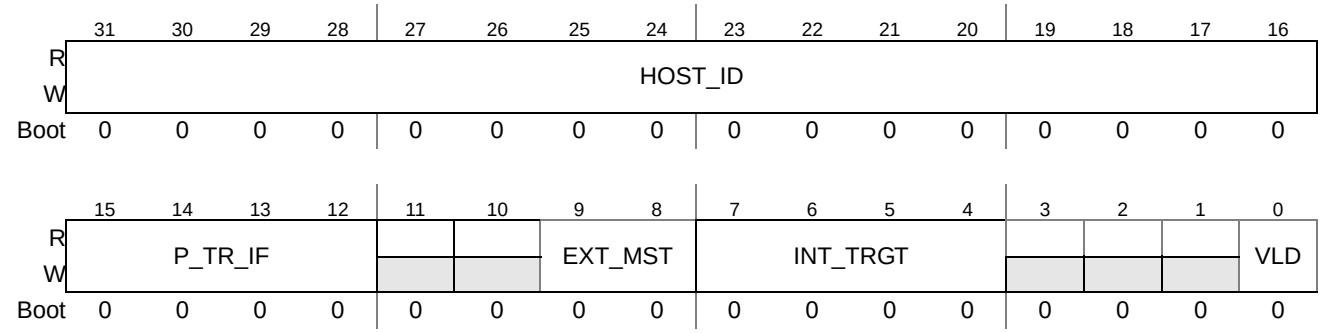


Figure 26-56. MAPLE DFTPE BD Ring Low Priority B <x> Parameter

These parameters, along with **MDFBRLAxP,** describe the attributes of a Low-Priority DFTPE BD ring. There are 8 such parameters, one for each possible Low-Priority DFTPE BD ring.

Table 26-82. MAPLE DFTPE BD Ring Low Priority B <x> Parameter Fields Description

Field	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by the Serial RapidIO port and owner of this ring. This field is valid only if MDFBRLPBxP[EXT_MST] equals 0b01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	

Table 26-82. MAPLE DFTPE BD Ring Low Priority B <x> Parameter Fields Description

Field	Description	Settings
EXT_MST 9–8	**External Master** Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*.	0b00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MDFBRLPBxP[INT_TRGT]. 0b01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated using MDFBRLPBxP[HOST_ID] (**Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*). 0b10 Reserved. 0b11 Reserved.
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MDFBRLPBxP[EXT_MST] equals 0b00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	
VLD 0	**Valid Bit** Describe if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.1.16 MAPLE CRCPE BD Ring High Priority A <x> Parameter (MCRCBRHPAxP)

Offset 0x00000200 (MCRCBRHPA0P) Access: User read/write
offset x*0x8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:5]													
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-57. MAPLE CRCPE BD Ring High Priority A <x> Parameter

These parameters, along with **MCRCBRHPBxP,** describe the attributes of a High-Priority CRCPE BD ring. There are 8 such parameters, one for each possible High-Priority CRCPE BD ring.

Table 26-83. MAPLE CRCPE BD Ring High Priority A <x> Parameter Fields Description

Name	Description	Settings
— 31–30	Reserved	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM. The given address is relative to the start address of the BD memory in the DRAM.	0x00 to 0x2F BD Ring Base Address for CRCPE High Priority Ring x. (stands for base address 0x0000 to 0x2F00
— 23–14	Reserved	
BDR_RD_PTR 13–5	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-increment by the MAPLE-B, after executing the BD. The host can use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA (BDR_RD_PTR[13:6] = {BDR_BA[13:8],0,0}), and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results. The address must be aligned to 16 bytes, which is the CRCPE BD size.	The pointer should be relative to the start address of the BD memory in the DRAM, therefore the range of addresses should be:0x000 to 0x7FF Read Pointer for CRCPE High Priority Ring x
— 4–0	Reserved	

26.4.2.1.17 MAPLE CRCPE BD Ring High Priority B <x> Parameter (MCRCBRHPBxP)

Offset 0x00000204(MCRCBRHPB0P) Access: User read/write
offset x*0x8
range x = 0...7

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	P_TR_IF						EXT_MST		INT_TRGT							VLD
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-58. MAPLE CRCPE BD Ring High Priority B <x> Parameter

These parameters, along with **MCRCBRHAxP,** describe the High-Priority CRCPE BD ring attributes. There are 8 such parameters, one for each possible High-Priority CRCPE BD ring.

Table 26-84. MAPLE CRCPE BD Ring High Priority B <x> Parameter Fields Description

Name	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by the Serial RapidIO port and owner of this ring. This field is valid only if MDFBRHPBxP[EXT_MST] equals 01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	
EXT_MST 9–8	**External Master** Describes whether the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3, *External Masters Support Using Serial RapidIO Doorbell***."	00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MCRCBRHPBxP[INT_TRGT]. 01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated (**Section 26.3.3.3, *External Masters Support Using Serial RapidIO Doorbell***"). 10 Reserved. 11 Reserved
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MCRCBRHPBxP[EXT_MST] equals 00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	
VLD 0	**Valid Bit** Describes if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared. 0 BD ring is not valid. 1 BD ring is valid.	

26.4.2.1.18 MAPLE CRCPE BD Ring Low Priority A <x> Parameter (MCRCBRLPAxP)

Offset 0x00000240 (MCRCBRLPA0P) Access: User read/write
offset x*0x8
range x = 0...7

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			BDR_BA[13:8]													
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			BDR_RD_PTR[13:5]													
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-59. MAPLE CRCPE BD Ring Low Priority A <x> Parameter

These parameters, along with **MCRCBRLBxP,** describe the attributes of a Low-Priority CRCPE BD ring. There are 8 such parameters, one for each possible Low-Priority CRCPE BD ring.

Table 26-85. MAPLE CRCPE BD Ring Low Priority A <x> Parameter Fields Description

Name	Description	Settings
— 31–30	Reserved.	
BDR_BA[13:8] 29–24	**BD Ring Base Address** Points to the base address of the BD ring in the PSIF DRAM. The address must be aligned to 256 bytes, and therefore, the field range is 13:8. The B.A should be within the 12 Kbytes of BD memory in the DRAM. The given address is relative to the start address of the BD memory in the DRAM.	0x00 to 0x2F—BD Ring Base Address for CRCPE Low Priority Ring x. Stands for base address 0x0000 to 0x2F00.
— 23–14	Reserved	
BDR_RD_PTR[13:5] 13–3	**BD Ring Read Pointer** This pointer points to the current BD in the ring the MAPLE-B is currently processing. It is post-increment by the MAPLE-B, after executing the BD. The host can use this pointer to track the MAPLE-B progress in the BD ring. This read pointer must be initialized once by the host when initializing the BDR_BA field, and with the same value as BDR_BA (BDR_RD_PTR[13:6] = {BDR_BA[13:8],0,0}), and must not be overwritten once the operation starts. Altering the contents of this pointer by the host will have unexpected results. The address must be aligned to 16 bytes, which is the CRCPE BD size.	The pointer should be relative to the start address of the BD memory in the DRAM, therefore the range of addresses should be: 0x000 to 0x7FF—Read Pointer for CRCPE Low Priority Ring x.
— 2–0	Reserved	

26.4.2.1.19 MAPLE CRCPE BD Ring Low Priority B <x> Parameter (MCRCBRLPBxP)

Offset 0x00000244(MCRCBRLPB0P) Access: User read/write
offset x*0x8
range x = 0...7

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	P_TR_IF						EXT_MST		INT_TRGT							VLD
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-60. MAPLE CRCPE BD Ring Low Priority B <x> Parameter

These parameters, along with MCRCBRLAxP, describe the attributes of a Low-Priority CRCPE BD ring. There are 8 such parameters, one for each possible Low-Priority CRCPE BD ring.

Table 26-86. MAPLE CRCPE BD Ring Low Priority B <x> Parameter Field Description

Name	Description	Settings
HOST_ID 31–16	**16 bits of Host ID** For Serial RapidIO door-bell support for an external (to SoC) master connected to the SoC by the Serial RapidIO port and owner of this ring. This field is valid only if MCRCBRLPBxP[EXT_MST] equals 01.	
P_TR_IF 15–12	**Port Target Interface** Determines which RapidIO port to use for this doorbell. This field is used by the MAPLE-B for the ODDATR[TGINT] field in the Serial RapidIO Doorbell controller.	
— 11–10	Reserved	
EXT_MST 9–8	**External Master** Describes if the Ring owner can receive any of the MAPLE-B regular interrupt lines, and a regular interrupt (one out of 16) is generated if the [INT_EN] bit in the BD is set; or if the ring owner is connected to the SoC via Serial RapidIO port and a Serial RapidIO door-bell interrupt is generated if the [INT_EN] bit in the BD is set. For more details see **Section 26.3.3.3, *External Masters Support Using Serial RapidIO Doorbell***."	00 The master of this ring can receive any of the MAPLE-B regular interrupts, therefore a regular interrupt is generated according to MCRCBRLPBxP[INT_TRGT]. 01 The master of this ring is external (connected via Serial RapidIO port), therefore a Serial RapidIO door-bell interrupt is generated (**Section 26.3.3.3, *External Masters Support Using Serial RapidIO Doorbell***"). 10 Reserved. 11 Reserved.
INT_TRGT 7–4	**Target Interrupt** Defines which regular interrupt is to be asserted due to task completion in this BD ring. This field is valid only if MCRCBRLPBxP[EXT_MST] equals 00.	0000 IRQ0 0001 IRQ1 ... 1111 IRQ15
— 3–1	Reserved	

Table 26-86. MAPLE CRCPE BD Ring Low Priority B <x> Parameter Field Description

Name	Description	Settings
VLD 0	**Valid Bit** Describes if the current BD ring is valid. The MAPLE-B uses this bit to determine the BD Ring validation, therefore any change in any of the BD Ring parameters must be done while the VLD bit is cleared.	0 BD ring is not valid. 1 BD ring is valid.

26.4.2.2 MAPLE Operating Parameters

26.4.2.2.1 MAPLE- Turbo Stop Criteria Configuration Parameter(MTSCCP)

Offset 0x000000300 (MTSCCP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									STOP_CNT						S_CRC	S_APP
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-61. MAPLE Turbo Stop Criteria Configuration Parameter

This parameter controls the Turbo stopping criteria parameters.

Table 26-87. MAPLE Turbo Stop Criteria Configuration Parameter Fields Description

Field	Description	Settings
— 31–8	Reserved	
STOP_CNT 7–4	**Stop Count** This value determines the threshold dedicated counter stop condition if the Aposteriori Quality Stop Criteria bit is enabled (S_APP). The decoding is stopped if the following condition is met in between iterations: $(\frac{\text{STOPCNT}}{16} \times \text{Num of Steps}) > \text{Number of Aposteriori results which is bigger than TVAQCR[AQTH]}$ "Num of Steps" is the number of Trellis steps in the decoded block. For the 3Gxx operation modes (binary decoding), It equals the block size + 3 (Zero Tail). For the WiMAX operation mode (Duo Binary decoding), the "Num of Steps" is multiplied by 2 because each actual trellis step represents two decoded bits.	
3–2	Reserved	
S_CRC 1	**CRC Stop Criteria** For details see **Section 26.3.2.1.1.5**, *Stopping Criteria*.	0 Disabled 1 Enabled
S_APP 0	**Aposteriori Quality Stop Criteria** For details see **Section 26.3.2.1.1.5**, *Stopping Criteria*.	0 Disabled 1 Enabled

Note: You cannot enable both stopping criteria at the same time. For each configuration, only one stopping criteria can be enabled.

26.4.2.2.2 MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter (MTVPVxHCP)

Offset 0x000000380 (MTVPVxHCP) Access: User read/write
offset 8
range x = 0...9

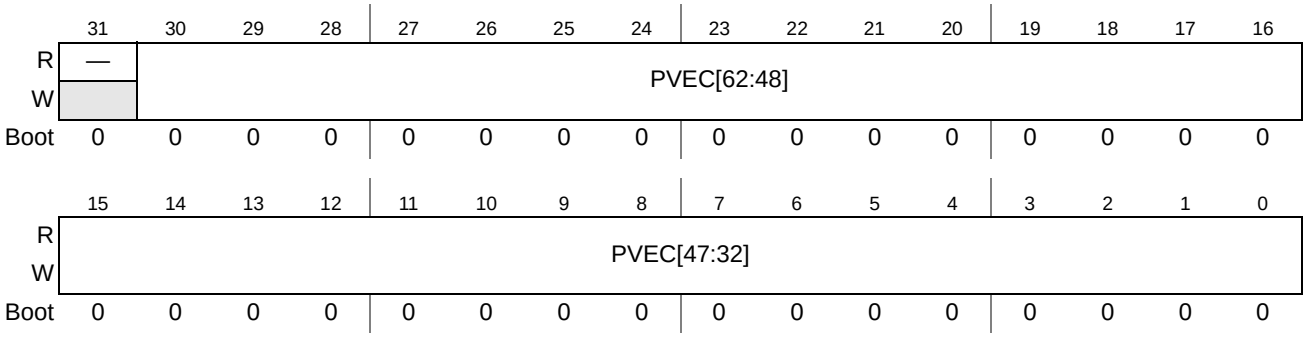


Figure 26-62. MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter

These parameters, together with the MTVPVxLCP parameters, control the Turbo Viterbi Puncturing Vectors configuration values related to the TVPE.

Table 26-88. MAPLE Turbo Viterbi Puncturing Vector x High Configuration Parameter Fields Description

Field	Description	Settings
— 31	Reserved.	
PVEC[62:32] 30–0	**Puncturing Vector** Last 32 bits of the puncturing vector.	For each bit in the vector: 0 Punctured symbol 1 Valid symbol

26.4.2.2.3 MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter (MTVPVxLCP)

Offset 0x000000384 (MTVPVxLCP) Access: User read/write
offset 8
range x = 0...9

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PVEC[31:16]															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PVEC[15:0]															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-63. MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter

These parameters, together with the MTVPVxHCP parameters, control the Turbo Viterbi Puncturing Vectors configuration values related to the TVPE.

Table 26-89. MAPLE Turbo Viterbi Puncturing Vector x Low Configuration Parameter Fields Description

Field	Description	
PVEC[31:0] 31–0	**Puncturing Vector** First 32 bits of the puncturing vector.	For each bit in the vector: 0 Punctured symbol 1 Valid symbol

26.4.2.2.4 MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter (MTVPPCyP)

Offset 0x0000003D0 (MTVPPCyP) Access: User read/write
offset 4
range y= 0...2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			PPER<0 + 4y>								PPER<1 + 4y>					
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			PPER<2 + 4y> (0 ≤ y ≤ 1)								PPER<3 + 4y> (0 ≤ y ≤ 1)					
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-64. MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter

These parameters contain the period of the puncturing vectors as described in the MTVPVxHCR and MTVPVxLCR.

Table 26-90. MAPLE Turbo Viterbi Puncturing Period Configuration y Parameter Fields Description

Field	Description	Settings
— 31–30	Reserved	
PPER<4y> 29–24	**Puncturing Period** This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.3.1.4.4**, *TVPE Input Data Structures*.	1–63 possible values which stand for the possible periods
— 23–22	Reserved	
PPER<4y+1> 21–16	**Puncturing Period** This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.3.1.4.4**, *TVPE Input Data Structures*.	1–63 possible values which stand for the possible periods
— 15–14	Reserved	
PPER<4y+2> 13–8	**Puncturing Period** This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.3.1.4.4**, *TVPE Input Data Structures*.	1–63 possible values which stand for the possible periods
— 7–6	Reserved	
PPER<4y+3> 5–0	**Puncturing Period** This field indicates the period of the puncturing if periodic puncturing is enabled. See **Section 26.3.1.4.4**, *TVPE Input Data Structures*.	1–63 possible values which stand for the possible periods

Note: The order in which the PVEC[62:0] vector is read is from bit 0] to bit 62] (MTVPVxLCP[0] is the first bit and MTVPVx(H/L)CP[PPER] is the last bit in the punctured pattern), where PPER is defined in MTVPPCyP.

26.4.2.2.5 MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 Parameter (MTVPVSxC0P)

Offset 0x0000003E0 (MTVPVSxC0P) Access: User read/write
offset 8
range x= 1...3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									M_POLYGEN1							
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									M_POLYGEN0							
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-65. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 Parameter

These parameters, along with the MTVPVSxC1P parameters, define three sets of Viterbi polynomial configurations. The MAPLE-B determines with which set to configure the TVPE (see **Section 26.4.4.1.6, *TVPE Viterbi Polynomial Vector Generation 0 Configuration Register (TVVPVG0CR)***) according to the [VIT_SET] field of the TVPE BD. For more details refer to **Section 26.3.1.4.2, *TVPE Buffer-Descriptor Structure***.

Table 26-91. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 0 Parameter Fields Description

Name	Description
— 31–24	Reserved
M_POLYGEN1 23–16	Field description is identical to the POLYGEN1 field description in **Table 26-117** on page 183.
— 15–8	Reserved
M_POLYGEN0 7–0	Field description is identical to the POLYGEN0 field description in **Table 26-118** on page 184

26.4.2.2.6 MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 Parameter (MTVPVSxC1P)

Offset 0x0000003E4 (MTVPVSxC1P) offset 8 range x= 1...3

Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R									M_POLYGEN3							
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									M_POLYGEN2							
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-66. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 Parameter

These parameters, along with the MTVPVSxC0P parameters, define three sets of Viterbi polynomial configurations. The MAPLE-B determines with which set to configure the TVPE (see **Section 26.4.4.1.7, *TVPE Viterbi Polynomial Vector Generation 1 Configuration Register (TVVPVG1CR)***) according to the [VIT_SET] field of the TVPE BD. For more details refer to **Section 26.3.1.4.2, *TVPE Buffer-Descriptor Structure***.

Table 26-92. MAPLE Turbo Viterbi Polynomial Vector Set x Configuration 1 Parameter Fields Description

Name	Description
— 31–24	Reserved
M_POLYGEN3 23–16	Field description is identical to the POLYGEN3 field description in **Table 26-118** on page 184.
— 15–8	Reserved
M_POLYGEN2 7–0	Field description is identical to the POLYGEN2 field description in **Table 26-118** on page 184

26.4.2.3 Profiling Parameters

26.4.2.3.1 MAPLE-B Turbo Total Performance Parameter (MTTPP)

Offset 0x000000440 (MTTPP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	T_PRF															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-67. MAPLE-B Turbo Total Performance Parameter

This parameter sums the total Turbo performance (sum of decoded bits) of the TVPE.

Table 26-93. MAPLE-B Turbo Total Performance Parameter Fields Description

Field	Description
T_PRF 31–0	**Turbo Performance** Sums the BS field values of the Turbo BDs which were executed by the TVPE (error due to wrong configuration will not be counted while CRC fail will be counted).

26.4.2.3.2 MAPLE-B Viterbi Total Performance Parameter (MVTPP)

Offset 0x000000444 (MVTPP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V_PRF															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-68. MAPLE-B Viterbi Total Performance Parameter

This parameter sums the total Viterbi performance (sum of decoded bits) of the TVPE.

Table 26-94. MAPLE-B Viterbi Total Performance Parameter Fields Description

Field	Description
V_PRF 31-0	**Viterbi Performance** Sums the BS field values of the completed Viterbi BDs.

26.4.2.3.3 MAPLE-B Total BLER Parameter (MTBP)

Offset 0x000000448 (MTBP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TBLER															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-69. MAPLE-B Total BLER Parameter

This parameter sums the total number of Turbo jobs which completed with CRC check pass indication.

Table 26-95. MAPLE-B Total BLER Parameter Fields Description

Field	Description
TBLER 31–0	**Total BLER** Once enabled, this parameter contain the sum of all the Turbo jobs which completed with CRC check indication passed. CRC pass indication can occur either by enabling CRC check ([CRC_EN] bit of the TVPE BD) or by enabling CRC stop criteria (MTSCCP[S_CRC] bit).

Note: Enabling the MTBP parameter does not mean that all Turbo jobs will be forced to run CRC check. If the CRC check ([CRC_EN] bit of the TVPE BD) or CRC stop criteria (MTSCCP[S_CRC] bit) are not enabled then the BLER counter is not increment.

26.4.2.3.4 MAPLE-B TVPE BDs Counter Parameter (MTBDCP)

Offset 0x00000044C (MTBDCP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	T_CMP															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-70. MAPLE-B TVPE BDs Counter Parameter

This parameter sums the total TVPE BDs which have completed.

Table 26-96. MAPLE-B TVPE BDs Counter Parameter Fields Description

Field	Description
T_CMP 31–0	**TVPE Complete** This counter sums the Total number of TVPE BDs which have completed since the counter was enabled.

26.4.2.3.5 MAPLE-B FFT Total Performance Parameter (MFTPP)

Offset 0x000000450 (MFTPP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F_PRF															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-71. MAPLE-B FFT Total Performance Parameter

This parameter sums the total FFT calculated samples of the FFTPE.

Table 26-97. MAPLE-B FFT Total Performance Parameter Fields Description

Field	Description
F_PRF 31–0	**FFT Performance** Sum of the total Transform Lengths of the completed FFT BDs.

26.4.2.3.6 MAPLE-B FFTPE BDs Counter Parameter (MFBDCP)

Offset 0x000000454 (MFBDCP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	F_CMP															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-72. MAPLE-B FFT BDs Counter Parameter

This parameter sums the total FFT BDs completed of the FFTPE.

Table 26-98. MAPLE-B FFT BDs Counter Parameter Fields Description

Field	Description
F_CMP 31–0	**FFTPE BDs Complete** Sum of the total number of completed FFTPE BDs.

26.4.2.3.7 MAPLE-B DFT Total Performance Parameter (MDTPP)

Offset 0x000000458 (MDTPP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_PRF															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-73. MAPLE-B DFT Total Performance Parameter

This parameter sums the total DFT calculated samples of the DFTPE.

Table 26-99. MAPLE-B DFT Total Performance Parameter Fields Description

Field	Description
D_PRF 31–0	**DFT Throughput** Sum of the total Transform Lengths of the completed DFT BDs.

26.4.2.3.8 MAPLE-B DFTPE BDs Counter Parameter (MDBDCP)

Offset 0x00000045C (MDBDCP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	D_CMP															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-74. MAPLE-B DFT BDs Counter Parameter

This parameter sums the total DFT BDs completed of the DFTPE.

Table 26-100. MAPLE-B DFT BDs Counter Parameter Fields Description

Field	Description
D_CMP 31–0	**DFTPE BDs Complete** Sum of the total number of completed DFTPE BDs.

26.4.2.3.9 MAPLE-B Profiling Enable Parameter (MPEP)

Offset 0x000000460 (MPEP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							MDBDCP_EN	MDTPP_EN			MFBDCP_EN	MFTPP_EN	MTBDCP_EN	MTBP_EN	MVTPP_EN	MTTPP_EN
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-75. MAPLE-B Profiling Enable Parameter

This parameter describes which profiling parameters are enabled on assertion of the toggle bit. See **Section 26.3.2.1.5**, *TVPE Debug and Profiling* and **Section 26.3.2.2.6**, *FTPE Profiling*.

Table 26-101. MAPLE-B Profiling Enable Parameter Fields Description

Field	Description	Settings
— 31–10	Reserved	
MDBDCP_EN 9	**MDBCDP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MDBDCP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
MDTPP_EN 8	**MDTPP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MDTPP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
— 7–6	Reserved	
MFBDCP_EN 5	**MFBDCP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MFBDCP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
MFTPP_EN 4	**MFTPP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MFTPP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
MTBDC_EN 3	**MDTPP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MDTPP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
MTBP_EN 2	**MTBP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MTBP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
MVTPP_EN 1	**MVTPP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MVTPP is enabled.	0 Parameter not enabled. 1 Parameter enabled.
MTTPP_EN 0	**MTTPP Enable** If set, on the next toggle bit assertion of the DFT BD that enables the profiling parameters, the MTTPP is enabled.	0 Parameter not enabled. 1 Parameter enabled.

26.4.2.4 Serial RapidIO Doorbell Support Attributes Parameters

See **Section 26.3.3.3**, *External Masters Support Using Serial RapidIO Doorbell*, on page 26-118 for the functional description. of the Serial RapidIO parameters.

26.4.2.4.1 Serial RapidIO Outbound RapidIO Doorbell Base Address Parameter (SORDP0BAP)

Offset 0x000000480 (SORDP0BAP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BA															
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-76. Serial RapidIO Outbound RapidIO Doorbell Base Address Parameter

This parameter defines the base address of the Serial RapidIO Outbound Doorbell controller for MAPLE-B.

Table 26-102. Serial RapidIO Outbound RapidIO Doorbell Base Address Parameter Fields Description

Field	Description
BA 31–0	**Base Address of the Outbound Doorbell Controller** Using this field MAPLE-B assumes the following addresses for the Outbound Doorbell controller: ODMR[1] BA + 0x0 ODSR BA + 0x4 ODDPR BA + 0x18 ODDATR BA + 0x1C ODRETCR BA + 0x2C This parameter is valid only if MBDR(H/L)PBxP[EXT_MST] equals 0b01.

1. For details on the Doorbell controller registers, see **Section 16.4.2**, *Doorbell Controller,* on page 16-109.

26.4.2.4.2 Hardware Semaphore Base Address Parameter (HSP0BAP)

Offset 0x000000488 (HSP0BAP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BA															HS_EN
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-77. Hardware Semaphore Base Address Parameter

This parameter defines the base address and the enable bit of the external Hardware Semaphore register to be used when accessing the Serial RapidIO Doorbell controller for MAPLE-B. If the enable bit is set, the Serial RapidIO Doorbell can only be accessed if the semaphore is locked by MAPLE-B.

Table 26-103. Hardware Semaphore Base Address Parameter Field Description

Field	Description	Settings
BA 31–2	**Base Address of the hardware Semaphore Register** If the HSP0BAP[HS_EN] is enabled and the MBDR(H/L)PBxP[EXT_MST] equals 0b01, then the MAPLE-B conditions its access of the Serial RapidIO Doorbell controller by locking the semaphore pointed to by this field. The address in this field must be 4 bytes aligned.	
— 1	Reserved.	
HS_EN 0	**Hardware Semaphore Enable** If set, the MAPLE-B must lock the external semaphore located in address HSP0BAP[BA] with the PE ID (MTGCP<pe>_HS_ID]) before it can access the doorbell controller.	0 The MAPLE-B uses internal semaphore implementation in order to arbitrate between the PEs trying to access the doorbell controller. 1 The MAPLE-B use external semaphore located at address HSP0BAP[BA] in order to arbitrate between the PEs and other external potential doorbell controller masters.

26.4.2.4.3 MAPLE-B Doorbell Hardware Semaphore ID Configuration Parameter (MDHSIDCP)

Offset 0x000000490 (MDHSIDCP) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CRC_HS_ID								TV_HS_ID							
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DF_HS_ID								FF_HS_ID							
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-78. MAPLE-B Hardware Semaphore ID

This parameter defines the ID number by which it locks the Hardware Semaphore register for MAPLE-B.

Table 26-104. MAPLE-B Hardware Semaphore ID Field Description

Name	Description
CRC_HS_ID 31–24	**CRCPE Hardware Semaphore ID** If the HSP(0/1)BAP[HS_EN] is enabled and the MCRCBR(H/L)PBxP[EXT_MST] equals ’01’/’10’, then the MAPLE-B must lock the External Hardware Semaphore (HS) before accessing the Serial RapidIO Doorbell controller. Accessing the HS for the CRCPE rings is done using the ID described in this field
TV_HS_ID 23–16	**TVPE Hardware Semaphore ID** If the HSP(0/1)BAP[HS_EN] is enabled and the MTVBR(H/L)PBxP[EXT_MST] equals ’01’/’10’, then the MAPLE-B must lock the External Hardware Semaphore (HS) before accessing the Serial RapidIO Doorbell controller. Accessing the HS for the TVPE rings is done using the ID described in this field.
DF_HS_ID 15–8	**DFTPE Hardware Semaphore ID** If the HSP(0/1)BAP[HS_EN] is enabled and the MDFBR(H/L)PBxP[EXT_MST] equals ’01’/’10’, then the MAPLE-B must lock the External Hardware Semaphore (HS) before accessing the Serial RapidIO Doorbell controller. Accessing the HS for the DFTPE rings is done using the ID described in this field.
FF_HS_ID 7–0	**FFTPE Hardware Semaphore ID** If the HSP(0/1)BAP[HS_EN] is enabled and the MFFBR(H/L)PBxP[EXT_MST] equals ’01’/’10’, then the MAPLE-B must lock the External Hardware Semaphore (HS) before accessing the Serial RapidIO Doorbell controller. Accessing the HS for the FFTPE rings is done using the ID described in this field.

26.4.2.4.4 MAPLE-B Doorbell General Configuration Parameter (MDGCP)

Offset 0x000000494 (MDGCP) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							BD_PR		RET							
W																
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-79. MAPLE-B Hardware Semaphore ID

This parameter defines general parameters to be used while accessing the serial RapidIO doorbell controller.

Table 26-105. MAPLE-B Hardware Semaphore ID Field Description

Name	Description
— 31–10	Reserved.
BD_PR 9–8	**Serial RapidIO Doorbell 'Transaction flow level'**. Defines the Priority of the doorbell transaction. This field is written by the MAPLE-B to the *OM0DATR[DTFLOWLVL]* register of the Doorbell controller.
RET 7–0	**Serial RapidIO Doorbell 'Retry error threshold'** Defines the number of times the Doorbell unit will attempt to transmit the Doorbell. This field is written by the MAPLE-B to the *ODRETCR[RET]* register of the Doorbell controller.

26.4.3 PSIF Registers

The PSIF registers are accessed via the MAPLE-B SBus port only.

26.4.3.1 PSIF Command Register (PCR)

Offset 0x00001000(PCR) Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											OPCODE					
W																
Boot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-80. PSIF Command Register

The PCR is the MAPLE-B command register. It is used for initializing specific MAPLE-B routines.

Table 26-106. PSIF Command Register Fields Description

Field	Description	Settings
— 31–17	Reserved.	
FLG 16	**Command Semaphore Flag** Set by the host and cleared by the MAPLE-B once execution is done. Once set, the MAPLE-B executes the requested routine as described in PCR[OPCODE], and cleared the FLG bit.	
— 16–6	Reserved.	
OPCODE 5–0	**OPCODE** On assertion of the PCR[FLG], the MAPLE-B executes the task encoded in this field according to the specified encoding: For details see **page 26-122**.	‘000000’ - Reserved. ‘000001’ - MAPLE_parse_bd. ‘000010’ - MAPLE_parse_tvpe_bd. ‘000011’ - MAPLE_parse_fftpe_bd. ‘000100’ - MAPLE_parse_dftpe_bd. ‘000101’ - MAPLE_parse_crcpe_bd. ‘000110’ to ‘111111’ - Reserved.

26.4.3.2 PSIF PIC Event Register (PSPICER)

Offset 0x00001600 (PSPICER) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				Res.	Res.	DP0_INT	Res.	Res.	TVPE_CNFG	DFTPE_CNFG	FFTPE_CNFG	Res.	Res.	Res.	Res.
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-81. PSIF PIC Event Register

This register describes which events are set. Each event can be set by the PSIF internal events. It is cleared only by the host writing a 1 to the relevant bit. The bits in this event register are set only if the interrupt configuration is ‘level’ (see **Section 26.4.3.3**, *PSIF PIC Edge/Level Register(PSPICELR)*, on page 26-171

Table 26-107. PSIF PIC Event Register Fields Description

Field	Description	Settings
— 31–26	Reserved	
DP0_INT 25	**Serial RapidIO Doorbell Interrupt Generation From Doorbell Controller Failed** This bit is set if the MAPLE-B received an error indication from Serial RapidIO doorbell controller on its doorbell generation. Will cause the assertion of the ipi_general_err interrupt line. Resetting the event and the ipi_general_err interrupt line is done by writing ‘1’ to the bit[1]	0 No error. 1 Error indication received.
— 24–23	Reserved	
TVPE_CNFG 22	**TVPE BD Configuration Error** Error occurred due to Wrong BS value or due to wrong IN_D_STRCT value which does not fit the chosen Turbo standard. Will cause the assertion of the ipi_general_err interrupt line. Resetting the event and the ipi_general_err interrupt line is done by writing ‘1’ to the bit[1]	0 No error. 1 Error indication received.
DFTPE_CNFG 21	**DFTPE BD Configuration Error** Error occurred due to non-valid TL_ID configuration in the DFTPE BD. Will cause the assertion of the ipi_general_err interrupt line. Resetting the event and the ipi_general_err interrupt line is done by writing ‘1’ to the bit[1]	0 No error. 1 Error indication received.
DFTPE_CNFG 20	**DFTPE BD Configuration Error** Error occurred due to non-valid TL_ID configuration in the FFTPE BD. Will cause the assertion of the ipi_general_err interrupt line. Resetting the event and the ipi_general_err interrupt line is done by writing ‘1’ to the bit[1]	0 No error. 1 Error indication received.
— 19–16	Reserved	

Table 26-107. PSIF PIC Event Register Fields Description (Continued)

Field	Description	Settings
BDR_DONE[15:0] 15–0	**Buffer Descriptor Done** 16 interrupt lines which can be asserted due to Buffer Descriptors completion. Each of these interrupts can be assigned to any of the BD rings, and can be configured as edge or level. For more details see **Section 26.4.2.1.4**, *MAPLE TVPE BD Ring High Priority A <x> Parameter (MTVBRHPAxP),* on page 26-134 to **Section 26.4.2.1.15**, *MAPLE DFTPE BD Ring Low Priority B <x> Parameter (MDFBRLPBxP),* on page 26-147. Assertion of these bits cause the assertion of the relevant ipi_bdr_done[15:0] interrupt line. These bits are asserted only if the relevant bit in the PSPICELR[15:0] is reset (‘level configuration). Resetting the events is done by writing ‘1’ to the bits	

26.4.3.3 PSIF PIC Edge/Level Register(PSPICELR)

Offset 0x00001604 (PSPICELR) Access: User read/write

	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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BDR_DONE_EL[15:0]															
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 26-82. PSIF PIC Edge/Level Register

For each interrupt, this register describes whether it is output as an edge or level interrupt.

Table 26-108. PSIF PIC Edge/Level Register Field Description

Field	Description	Settings
— 31–16	Reserved	
BDR_DONE_EL[15:0] 15–0	**BDR_DONE Interrupt Edge/Level Indication** Each bit in this bus fits its event in the PSPICER[15:0] register.	For each of these bits: 0 The interrupt assertion is of level type 1 The interrupt assertion is of edge type.

26.4.3.4 PSIF PIC Mask Register(PSPICMR)

Offset 0x00001608 (PSPICMR) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				RES.	RES.	DP0_INT_M	TVCOLDE_M	TVEXTCE_M	TVPE_CNFG_M	DFTPE_CNFG_M	FFTPE_CNFG_M	RES.	RES.	RES.	RES.
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-83. PSIF PIC Mask Register

This register describes the mask bit for each possible event.

Table 26-109. PSIF PIC Mask Register Fields Description

Field	Description	Settings
— 31–26	Reserved	
DP0_INT_M 25	**Doorbell controller event Mask indication**	0 The ipi_general_err interrupt is masked and will not assert due to DP0_INT event assertion. 1 The ipi_general_err interrupt is enabled and an interrupt will be asserted upon DP0_INT event assertion
TVCOLDE_M 24	**Collision on CD Memory Mask Indication**	0 The ipi_general_err interrupt is masked and will not assert due to TVCOLDE event assertion. 1 The ipi_general_err interrupt is enabled and an interrupt will be asserted upon TVCOLDE event assertion
TVEXTCE_M 23	**TVPE Extrinsic Data Collision Event Mask indication**	0 The ipi_general_err interrupt is masked and will not assert due to TVEXTCE event assertion. 1 The ipi_general_err interrupt is enabled and an interrupt will be asserted upon TVEXTCE event assertion
TVPE_CNFG_M 22	**TVPE BD Configuration eRror Event Mask Indication**	0 The ipi_general_err interrupt is masked and will not assert due to TVPE_CNFG event assertion. 1 The ipi_general_err interrupt is enabled and an interrupt will be asserted upon TVPE_CNFG event assertion

Table 26-109. PSIF PIC Mask Register Fields Description (Continued)

Field	Description	Settings
DFTPE_CNFG_M 21	**DFTPE BD Configuration Error Event Mask Indication**	0 The ipi_general_err interrupt is masked and will not assert due to DFTPE_CNFG event assertion. 1 The ipi_general_err interrupt is enabled and an interrupt will be asserted upon DFTPE_CNFG event assertion
FFTPE_CNFG_M 20	**FFTPE BD Configuration Error Event Mask Indication**	0 The ipi_general_err interrupt is masked and will not assert due to FFTPE_CNFG event assertion. 1 The ipi_general_err interrupt is enabled and an interrupt will be asserted upon FFTPE_CNFG event assertion
— 19-16	Reserved	
BDR_DONE_M[15:0] 15–0	**Buffer Descriptor Done Events Mask Indication**	For each of the bits: 0 The ipi_bdr_done[x] interrupt is masked and will not assert due to BDR_DONE[x] event assertion. 1 The ipi_bdr_done[x] interrupt is enabled and an interrupt will be asserted upon BDR_DONE[x] event assertion

26.4.3.5 PSIF PIC Interrupts Assertion Clocks Register (PSPICIACR)

Offset 0x0000160C (PSPICIACR) Access: User read/write

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

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

Figure 26-84. PSIF PIC Interrupt Assertion Clocks Register

For all the interrupts, this register describes the number of PSIF clock cycles the interrupt is asserted in case it is configured as an edge interrupt.

Table 26-110. mp_pic Interrupt Assertion Clocks Register Field Description

Field	Description	Settings
31–2	Reserved	
IAC 1–0	**Interrupt Assertions Clocks** Describe the number of clock cycles the interrupts are asserted if they are configured as edge interrupts.	00 The edge interrupt is asserted for 2 PSIF clock cycle. 11 The edge interrupt is asserted for 4 PSIF clock cycles. 10 The edge interrupt is asserted for 6 PSIF clock cycles. 11 The edge interrupt is asserted for 8 PSIF clock cycles.

26.4.4 Processing Engine Registers

26.4.4.1 TVPE Registers

The TVPE Register is accessed via the MBus slave interface.

26.4.4.1.1 TVPE Configuration 0 Register (TVPEC0R)

Offset 0x0005E000 Access: User read/write

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														Res.	DOSBY	DOSBI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0

Figure 26-85. TVPE Configuration 0 Register

This register contains the general configuration parameters for the TVPE. All reserved bits must be cleared to 0.

Table 26-111. TVPE Configuration 0 Fields Description

Field	Description	Settings
— 31–3	Reserved	
— 2	Reserved	
DOSBY 1	**Data Out Byte Order**	0 Descending order 1 Ascending order
DOSBI 0	**Data Out Bit Order**	0 Descending order 1 Ascending order

26.4.4.1.2 TVPE Symbol Identification 0 Configuration Register (TVSI0CR)

Offset 0x0005E008 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SID0				SID1				SID2				SID3			
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SID4				SID5				SID6				SID7			
Reset	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1

Figure 26-86. TVPE Symbol Identification 0 Configuration Register

This register, along with the TVSI1CR register, allows permutation for the default Symbol-Identification used for the different input data structures. For details, see **Section 26.3.1.4.4**, *TVPE Input Data Structures*, on page 26-24.

Table 26-112. TVPE Symbol Identification 0 Configuration Fields Description

Field	Description	Settings
SID0 31–28	**Symbol 0** Symbol Identification of the 1st channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is the 1st symbol in the default stream. 0001—Symbol 1 is permuted as symbol No. 0 0010—Symbol 2 is permuted as symbol No. 0 0011—Symbol 3 is permuted as symbol No. 0 0100—Symbol 4 is permuted as symbol No. 0 0101—Symbol 5 is permuted as symbol No. 0 0110—Symbol 6 is permuted as symbol No. 0 0111—Symbol 7 is permuted as symbol No. 0 1000—Symbol 8 is permuted as symbol No. 0 1001—Symbol 9 is permuted as symbol No. 0 1010 to 1111—Reserved.
SID1 27– 24	**Symbol 1** Symbol Identification of the 2nd channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 1 0001—Symbol 1 is the 2nd symbol in the default stream. 0010—Symbol 2 is permuted as symbol No. 1 0011—Symbol 3 is permuted as symbol No. 1 0100—Symbol 4 is permuted as symbol No. 1 0101—Symbol 5 is permuted as symbol No. 1 0110—Symbol 6 is permuted as symbol No. 1 0111—Symbol 7 is permuted as symbol No. 1 1000—Symbol 8 is permuted as symbol No. 1 1001—Symbol 9 is permuted as symbol No. 1 1010 to 1111—Reserved.

Table 26-112. TVPE Symbol Identification 0 Configuration Fields Description (Continued)

Field	Description	Settings
SID2 23–20	**Symbol 2** Symbol Identification of the 3rd channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 2 0001—Symbol 1 is permuted as symbol No. 2 0010—Symbol 2 is the 3rd symbol in the default stream. 0011—Symbol 3 is permuted as symbol No. 2 0100—Symbol 4 is permuted as symbol No. 2 0101—Symbol 5 is permuted as symbol No. 2 0110—Symbol 6 is permuted as symbol No. 2 0111—Symbol 7 is permuted as symbol No. 2 1000—Symbol 8 is permuted as symbol No. 2 1001—Symbol 9 is permuted as symbol No. 2 1010 to 1111—Reserved.
SID3 19–16	**Symbol 3** Symbol Identification of the 4th channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 3 0001—Symbol 1 is permuted as symbol No. 3 0010—Symbol 2 is permuted as symbol No. 3 0011—Symbol 3 is the 4th symbol in the default stream. 0100—Symbol 4 is permuted as symbol No. 3 0101—Symbol 5 is permuted as symbol No. 3 0110—Symbol 6 is permuted as symbol No. 3 0111—Symbol 7 is permuted as symbol No. 3 1000—Symbol 8 is permuted as symbol No. 3 1001—Symbol 9 is permuted as symbol No. 3 1010 to 1111—Reserved.
SID4 15– 12	**Symbol 4** Symbol Identification of the 5th channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 4 0001—Symbol 1 is permuted as symbol No. 4 0010—Symbol 2 is permuted as symbol No. 4 0011—Symbol 3 is permuted as symbol No. 4 0100—Symbol 4 is the 5th symbol in the default stream. 0101—Symbol 5 is permuted as symbol No. 4 0110—Symbol 6 is permuted as symbol No. 4 0111—Symbol 7 is permuted as symbol No. 4 1000—Symbol 8 is permuted as symbol No. 4 1001—Symbol 9 is permuted as symbol No. 4 1010 to 1111—Reserved.
SID5 11–8	**Symbol 5** Symbol Identification of the 6th channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 5 0001—Symbol 1 is permuted as symbol No. 5 0010—Symbol 2 is permuted as symbol No. 5 0011—Symbol 3 is permuted as symbol No. 5 0100—Symbol 4 is permuted as symbol No. 5 0101—Symbol 5 is the 6th symbol in the default stream. 0110—Symbol 6 is permuted as symbol No. 5 0111—Symbol 7 is permuted as symbol No. 5 1000—Symbol 8 is permuted as symbol No. 5 1001—Symbol 9 is permuted as symbol No. 5 1010 to 1111—Reserved.

Table 26-112. TVPE Symbol Identification 0 Configuration Fields Description (Continued)

Field	Description	Settings
SID6 7–4	**Symbol 6** Symbol Identification of the 7th channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 6 0001—Symbol 1 is permuted as symbol No. 6 0010—Symbol 2 is permuted as symbol No. 6 0011—Symbol 3 is permuted as symbol No. 6 0100—Symbol 4 is permuted as symbol No. 6 0101—Symbol 5 is permuted as symbol No. 6 0110—Symbol 6 is the 7th symbol in the default stream. 0111—Symbol 7 is permuted as symbol No. 6 1000—Symbol 8 is permuted as symbol No. 6 1001—Symbol 9 is permuted as symbol No. 6 1010 to 1111—Reserved.
SID7 3–0	**Symbol 7** Symbol Identification of the 8th channel data byte in the periodic stream (after de-puncturing).	0000—Symbol 0 is permuted as symbol No. 7 0001—Symbol 1 is permuted as symbol No. 7 0010—Symbol 2 is permuted as symbol No. 7 0011—Symbol 3 is permuted as symbol No. 7 0100—Symbol 4 is permuted as symbol No. 7 0101—Symbol 5 is permuted as symbol No. 7 0110—Symbol 6 is permuted as symbol No. 7 0111—Symbol 7 is the 8th symbol in the default stream. 1000—Symbol 8 is permuted as symbol No. 7 1001—Symbol 9 is permuted as symbol No. 7 1010 to 1111—Reserved.

Table 26-113 describes the assumed default streams for all possible use cases of the PPCMS input data structure:

Table 26-113. Default Symbol Ordering in PPCMS

Operation Mode	Rate	Default Symbol Ordering
All (Viterbi)	1/2	P0[n], P1[n], P0[n+1], P1[n+1], P0[n+2], P1[n+2], P0[n+3], P1[n+3]
All (Viterbi)	1/3	P0[n], P1[n], P2[n], P0[n+1], P1[n+1], P2[n+1]
All (Viterbi)	1/4	P0[n], P1[n], P2[n], P3[n], P0[n+1], P1[n+1], P2[n+1], P3[n+1]

26.4.4.1.3 TVPE Symbol Identification 1 Configuration Register (TVSI1CR)

Offset 0x0005E00C Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SID8				SID9											
W																
Reset	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0

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

Figure 26-87. TVPE Symbol Identification 1 Configuration Register

This register, along with the TVSI0CR register, allows permutation for the default Symbol-Identification used for the different input data structures. For details, see **Section 26.3.1.4.4**, *TVPE Input Data Structures*.

Table 26-114. TVPE Symbol Identification 1 Configuration Fields Description

Field	Description	Settings
SID8 31–28	**Symbol 8** Symbol Identification of the 9th channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 8 0001—Symbol 1 is permuted as symbol No. 8 0010—Symbol 2 is permuted as symbol No. 8 0011—Symbol 3 is permuted as symbol No. 8 0100—Symbol 4 is permuted as symbol No. 8 0101—Symbol 5 is permuted as symbol No. 8 0110—Symbol 6 is permuted as symbol No. 8 0111—Symbol 7 is permuted as symbol No. 8 1000—Symbol 8 is the 9th symbol in the default stream. 1001—Symbol 9 is permuted as symbol No. 8 1010 to 1111—Reserved.
SID9 27–24	**Symbol 9** Symbol Identification of the 10th channel data byte in the periodic stream (after de-puncturing)	0000—Symbol 0 is permuted as symbol No. 9 0001—Symbol 1 is permuted as symbol No. 9 0010—Symbol 2 is permuted as symbol No. 9 0011—Symbol 3 is permuted as symbol No. 9 0100—Symbol 4 is permuted as symbol No. 9 0101—Symbol 5 is permuted as symbol No. 9 0110—Symbol 6 is permuted as symbol No. 9 0111—Symbol 7 is permuted as symbol No. 9 1000—Symbol 8 is permuted as symbol No. 9 1001—Symbol 9 is the 10th symbol in the default stream. 1010 to 1111—Reserved.
— 23–0	Reserved.	

26.4.4.1.4 TVPE Turbo Tail Symbol Identification x Configuration Register (TVTTSIxCR)

Offset 0x0005E010 + 4x — Access: User read/write

x = 0...4 For X=4, only T16 and T17 exist

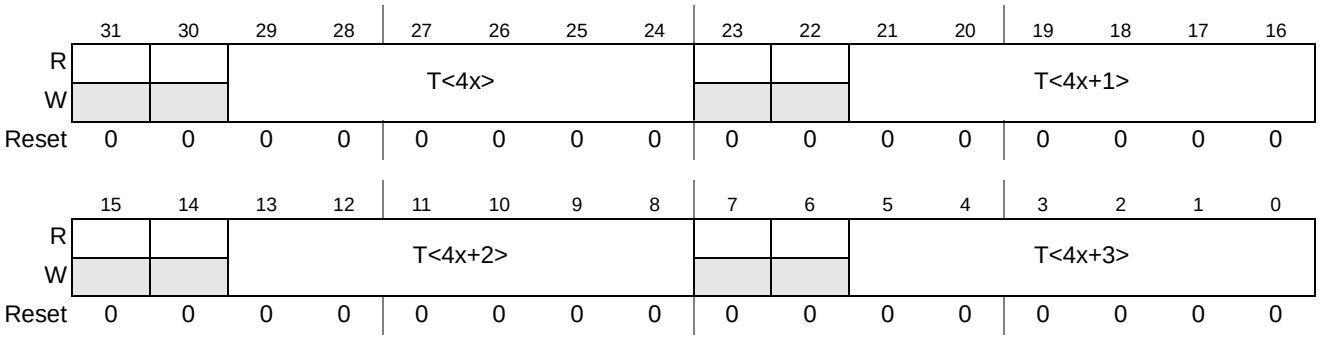


Figure 26-88. TVPE Turbo Tail Symbol Identification x Configuration Register

This group of registers identifies the tail symbols written through the channel data Stream Processor, when zero tail bytes is enabled ([ZTTB] bit of the TVPE BD is set). See **Section 26.3.1.4.4**, *TVPE Input Data Structures*, on page 26-24.

Table 26-115. TVPE Turbo Tail Symbol Identification X Fields Description

Field	Description	Settings
— 31–30	Reserved	
$T_{<4x>}$ 29–24	**First Tail Symbol** Symbol identification for the first symbol in the TVTTSIxCR register	0—SD(0) (The current tail symbol is the First SD symbol in the tail) 1—SD(1) (The current tail symbol is the Second SD symbol in the tail) 2—SD(2) (The current tail symbol is the Third SD symbol in the tail) 3—Reserved 4—PFa(0) (The current tail symbol is the First PFa symbol in the tail) 5—PFa(1) (The current tail symbol is the Second PFa symbol in the tail) 6—PFa(2) (The current tail symbol is the Third PFa symbol in the tail) 7— Reserved 8—PFb(0) (The current tail symbol is the First PFb symbol in the tail) 9—PFb(1) (The current tail symbol is the Second PFb symbol in the tail) 10—PFb(2) (The current tail symbol is the Third PFb symbol in the tail) 11—Reserved 12—PSa(0) (The current tail symbol is the First PSa symbol in the tail) 13—PSa(1) (The current tail symbol is the Second PSa symbol in the tail) 14—PSa(2) (The current tail symbol is the Third PSa symbol in the tail) 15—Reserved 16—PSb(0) (The current tail symbol is the First PSb symbol in the tail) 17—PSb(1) (The current tail symbol is the Second PSb symbol in the tail) 18—PSb(2) (The current tail symbol is the Third PSb symbol in the tail) 19—Reserved 20—SD~(0) (The current tail symbol is the First SD~ symbol in the tail) 21—SD~(1) (The current tail symbol is the Second SD~ symbol in the tail) 22—SD~(2) (The current tail symbol is the Third SD~ symbol in the tail) 23—Reserved 24 to 63 Reserved
— 23–22	Reserved	

Table 26-115. TVPE Turbo Tail Symbol Identification X Fields Description (Continued)

Field	Description	Settings
$T_{<4x+1>}$ 21–16	**Second Tail Symbol** Symbol identification for the second symbol in the TVTTSIxCR register	0—SD(0) (The current tail symbol is the First SD symbol in the tail) 1—SD(1) (The current tail symbol is the Second SD symbol in the tail) 2—SD(2) (The current tail symbol is the Third SD symbol in the tail) 3—Reserved 4—PFa(0) (The current tail symbol is the First PFa symbol in the tail) 5—PFa(1) (The current tail symbol is the Second PFa symbol in the tail) 6—PFa(2) (The current tail symbol is the Third PFa symbol in the tail) 7— Reserved 8—PFb(0) (The current tail symbol is the First PFb symbol in the tail) 9—PFb(1) (The current tail symbol is the Second PFb symbol in the tail) 10—PFb(2) (The current tail symbol is the Third PFb symbol in the tail) 11—Reserved 12—PSa(0) (The current tail symbol is the First PSa symbol in the tail) 13—PSa(1) (The current tail symbol is the Second PSa symbol in the tail) 14—PSa(2) (The current tail symbol is the Third PSa symbol in the tail) 15—Reserved 16—PSb(0) (The current tail symbol is the First PSb symbol in the tail) 17—PSb(1) (The current tail symbol is the Second PSb symbol in the tail) 18—PSb(2) (The current tail symbol is the Third PSb symbol in the tail) 19—Reserved 20—SD~(0) (The current tail symbol is the First SD~ symbol in the tail) 21—SD~(1) (The current tail symbol is the Second SD~ symbol in the tail) 22—SD~(2) (The current tail symbol is the Third SD~ symbol in the tail) 23—Reserved 24 to 63 Reserved
15–14	Reserved	
$T_{<4x+2>}$ 13–8	**Third Tail Symbol** Symbol identification for the third symbol in the TVTTSIxCR register	0—SD(0) (The current tail symbol is the First SD symbol in the tail) 1—SD(1) (The current tail symbol is the Second SD symbol in the tail) 2—SD(2) (The current tail symbol is the Third SD symbol in the tail) 3—Reserved 4—PFa(0) (The current tail symbol is the First PFa symbol in the tail) 5—PFa(1) (The current tail symbol is the Second PFa symbol in the tail) 6—PFa(2) (The current tail symbol is the Third PFa symbol in the tail) 7— Reserved 8—PFb(0) (The current tail symbol is the First PFb symbol in the tail) 9—PFb(1) (The current tail symbol is the Second PFb symbol in the tail) 10—PFb(2) (The current tail symbol is the Third PFb symbol in the tail) 11—Reserved 12—PSa(0) (The current tail symbol is the First PSa symbol in the tail) 13—PSa(1) (The current tail symbol is the Second PSa symbol in the tail) 14—PSa(2) (The current tail symbol is the Third PSa symbol in the tail) 15—Reserved 16—PSb(0) (The current tail symbol is the First PSb symbol in the tail) 17—PSb(1) (The current tail symbol is the Second PSb symbol in the tail) 18—PSb(2) (The current tail symbol is the Third PSb symbol in the tail) 19—Reserved 20—SD~(0) (The current tail symbol is the First SD~ symbol in the tail) 21—SD~(1) (The current tail symbol is the Second SD~ symbol in the tail) 22—SD~(2) (The current tail symbol is the Third SD~ symbol in the tail) 23—Reserved 24 to 63 Reserved
— 7–6	Reserved	

Table 26-115. TVPE Turbo Tail Symbol Identification X Fields Description (Continued)

Field	Description	Settings
$T_{<4x+3>}$ 5–0	**Fourth Tail Symbol** Symbol identification for the fourth symbol in the TVTTSIxCR register	0—SD(0) (The current tail symbol is the First SD symbol in the tail) 1—SD(1) (The current tail symbol is the Second SD symbol in the tail) 2—SD(2) (The current tail symbol is the Third SD symbol in the tail) 3—Reserved 4—PFa(0) (The current tail symbol is the First PFa symbol in the tail) 5—PFa(1) (The current tail symbol is the Second PFa symbol in the tail) 6—PFa(2) (The current tail symbol is the Third PFa symbol in the tail) 7— Reserved 8—PFb(0) (The current tail symbol is the First PFb symbol in the tail) 9—PFb(1) (The current tail symbol is the Second PFb symbol in the tail) 10—PFb(2) (The current tail symbol is the Third PFb symbol in the tail) 11—Reserved 12—PSa(0) (The current tail symbol is the First PSa symbol in the tail) 13—PSa(1) (The current tail symbol is the Second PSa symbol in the tail) 14—PSa(2) (The current tail symbol is the Third PSa symbol in the tail) 15—Reserved 16—PSb(0) (The current tail symbol is the First PSb symbol in the tail) 17—PSb(1) (The current tail symbol is the Second PSb symbol in the tail) 18—PSb(2) (The current tail symbol is the Third PSb symbol in the tail) 19—Reserved 20—SD~(0) (The current tail symbol is the First SD~ symbol in the tail) 21—SD~(1) (The current tail symbol is the Second SD~ symbol in the tail) 22—SD~(2) (The current tail symbol is the Third SD~ symbol in the tail) 23—Reserved 24 to 63 Reserved

26.4.4.1.5 TVPE Aposteriori Quality Configuration Register (TVAQCR)

Offset 0x0005E03C Access: User read/write

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

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

Figure 26-89. TVPE Aposteriori Quality Configuration Register

This register is used to describe the threshold above which the calculated aposteriori count is counted as good quality. It is used during Turbo processing only.

Table 26-116. TVPE Aposteriori Quality Configuration Fields Description

Field	Description
— 31–15	Reserved
AQTH 14–0	**Aposteriori Quality Threshold** For 3Gxx Turbo decoding (binary decoding): the number of the LLR values processed by the TVPE, whose absolute value is larger than this field value are counted. For WiMAX operation mode (duo-binary decoding): the number of Aposteriory values processed the TVPE, whose maximal value is greater than their counterpart Aposteriory values (The counterpart for 00 is 11, for 01 is 10, and so forth) are counted (as two). For details see **Section 26.3.2.1.1.5**, *Stopping Criteria,* on page 26-59.

26.4.4.1.6 TVPE Viterbi Polynomial Vector Generation 0 Configuration Register (TVVPVG0CR)

Offset 0x0005E040 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								POLYGEN1								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								POLYGEN0								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-90. TVPE Viterbi Polynomial Vector Generation 0 Configuration Register

This register is used with **TVVPVG1CR** to describe the generation of the polynomial for Viterbi decoding. To generate the parity bits, the following equation is used:

$Parity_p$ =

(new_encoded_bit & $POLYGEN_p[8]$) ^
(state[7] & $POLYGEN_p[7]$) ^
(state[6] & $POLYGEN_p[6]$) ^
(state[5] & $POLYGEN_p[5]$) ^
(state[4] & $POLYGEN_p[4]$) ^
(state[3] & $POLYGEN_p[3]$) ^
(state[2] & $POLYGEN_p[2]$) ^
(state[1] & $POLYGEN_p[1]$) ^
(state[0] & $POLYGEN_p[0]$)

Where “state” is the state register, composed of up to 8 FFs, in the Viterbi encoder, and “new_encoded_bit” is the currently encoded bit outside the “state” register.

Table 26-117. TVPE Viterbi Polynomial Vector Generation 0 Configuration Fields Description

Field	Description	Settings
— 31–25	Reserved	
POLYGEN1 24–16	This field enables bits for the generation of the second parity bit.	0 to (2^k – 1) possible values.
— 15–9	Reserved	
POLYGEN0 8–0	This field enables bits for the generation of the first parity bit.	0 to (2^k – 1) possible values.

26.4.4.1.7 TVPE Viterbi Polynomial Vector Generation 1 Configuration Register (TVVPVG1CR)

Offset 0x0005E044 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R								POLYGEN3								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R								POLYGEN2								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-91. TVPE Viterbi Polynomial Vector Generation 1 Configuration Register

This register, along with **TVVPVG0CR**, is used to describe the generation of the polynomial for Viterbi decoding. To generate the parity bits, the following equation is used:

$Parity_p$ =

(new_encoded_bit & $POLYGEN_p$[8]) ^
(state[7] & $POLYGEN_p$[7]) ^
(state[6] & $POLYGEN_p$[6]) ^
(state[5] & $POLYGEN_p$[5]) ^
(state[4] & $POLYGEN_p$[4]) ^
(state[3] & $POLYGEN_p$[3]) ^
(state[2] & $POLYGEN_p$[2]) ^
(state[1] & $POLYGEN_p$[1]) ^
(state[0] & $POLYGEN_p$[0])

Where “state” is the state register, composed of up to 8 FFs, in the Viterbi encoder, and “new_encoded_bit” is the currently encoded bit outside the “state” register.

Table 26-118. TVPE Viterbi Polynomial Vector Generation 1 Configuration Fields Description

Field	Description	Settings
— 31–25	Reserved	
POLYGEN3 24–16	This field enables bits for the generation of the fourth parity bit. Valid only for Viterbi Rate 1/4, must contain 0x0000 otherwise.	0 to (2^k – 1) possible values.
— 15–9	Reserved	
POLYGEN2 8–0	This field enables bits for the generation of the third parity bit. Valid only for Viterbi Rate 1/3 & 1/4, must contain 0x0000 otherwise.	0 to (2^k – 1) possible values.

26.4.4.1.8 TVPE Decoder Status Register (TVPESR)

Offset 0x0005ED00 Access: User read

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

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

Figure 26-92. TVPE Decoder Status Register

This register contains the TVPE TASKIDS status information.

Table 26-119. TVPE Decoder Status Fields Description

Field	Description
— 31–23	Reserved
— 22–16	Reserved
— 15–13	Reserved
— 12–8	Reserved
TASKIDS 7–0	**Task ID Status** The task ID of the command that is currently being executed or of the last command executed.

26.4.4.2 FFTPE Registers

26.4.4.2.1 FFTPE Data Size Register 0 (FFTPEDSR0)

Offset 0x0007E140 (FFTPEDSR0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DS0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DS1									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-93. FFTPE Data Size Register 0

This register records the data size for iFFTs (data size = [<transform length> – <guards size> – 1] / 4), as described by **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92.

Table 26-121. FFTPE Data Size Register 0 Field Descriptions

Field	Description
— 31–26	Reserved
DS0 25–17	**Data Size 0** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 128. The value of the DS0 should not exceed the 2≤ DS0 ≤ (128/4-1) values. For details, see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 16–10	Reserved
DS1 9–1	**Data Size 1** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 256. The value of the DS1 should not exceed the 2≤ DS1 ≤ (256/4-1) values. For details, see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 0	Reserved

26.4.4.2.2 FFTPE Data Size Register 1 (FFTPEDSR1)

Offset 0x0007E144 (FFTPEDSR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DS2									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DS3									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-94. FFTPE Data Size Register 1

This register records the data size for iFFTs (data size = [<transform length> – <guards size> – 1] / 4), as described by **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92.

Table 26-122. FFTPE Data Size Register 1 Field Descriptions

Field	Description
— 31–26	Reserved
DS2 25–17	**Data Size 2** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 512. The value of the DS2 should not exceed the 2≤ DS2 ≤ (512/4-1) values. For details, see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 16–10	Reserved
DS3 9–1	**Data Size 3** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 1024. The value of the DS3 should not exceed the 2≤ DS3 ≤ (1024/4-1) values. For details, see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 0	Reserved

26.4.4.2.3 FFTPE Data Size Register 2 (FFTPEDSR2)

Offset 0x0007E148 (FFTPEDSR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DS4									
W																
Reset																

	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	0	0	0	0	0
W																
Reset																

Figure 26-95. FFTPE Data Size Register 2

This register records the data size for iFFTs (data size = [<transform length> - <guards size> - 1] / 4), as described by **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92.

Table 26-123. FFTPE Data Size Register 2 Field Descriptions

Field	Description
— 31–26	Reserved
DS4 25–17	**Data Size 4** The actual size of the Right/Left symbol carrier in an iFFT process with Transform length of 2048. The value of the DS4 should not exceed the 2≤ DS4 ≤ (2048/4-1) values. For more details, see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 16–0	Reserved

26.4.4.2.4 FFTPE Status Register (FFTPESTR)

Offset 0x0007E238 (FFTPESTR) Access: Read

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TASKIDS								0	Res.			0	0	Res.	Res.
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-96. FFTPE Status Register

This register stores the Task ID of the current Transform written by MAPLE-B. When a new transform execution starts, this register is cleared by the FFTPE.

Table 26-124. FFTPE Status Register Field Descriptions

Field	Description
— 31–16	Reserved
TASKIDS 15–8	**Task ID Status** The task ID status of the command that is currently being executed or of the last command executed
— 7–0	Reserved

26.4.4.2.5 FFTPE Scaling Status Register (FFTPESCLSTR)

Offset 0x0007E23C (FFTPESCLSTR) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	SCL0			0	SCL1			0	SCL2			0	SCL3		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SCL4			0	SCL5			0	0	0	0	OVA_SCL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-97. FFTPE Scaling Status Register

This register describes the scaling amount applied at each stage during the Adaptive Scaling mode. It also describes the Overall Scaling amount applied during the whole transform processing.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-125. FFTPE Scaling Status Register Field Descriptions

Field	Description	Settings
— 31	Reserved	
SCL0 30–28	**Scale After Stage 0** The number of bits which have been scaled down(shift right) from all inputs after stage 0	Supported values: 0x0–0x4.
— 27	Reserved	
SCL1 26–24	**Scale After Stage 1** The number of bits which have been scaled down(shift right) from all inputs after stage 1	Supported values: 0x0–0x4.
— 23	Reserved	
SCL2 22–20	**Scale After Stage 2** The number of bits which have been scaled down(shift right) from all inputs after stage 2	Supported values: 0x0–0x4.
— 19	Reserved	
SCL3 18–16	**Scale After Stage 3** The number of bits which have been scaled down(shift right) from all inputs after stage 3	Supported values: 0x0–0x4.
— 15	Reserved	
SCL4 14–12	**Scale After Stage 4** The number of bits which have been scaled down(shift right) from all inputs after stage 4	Supported values: 0x0–0x4.
— 11	Reserved	

Table 26-125. FFTPE Scaling Status Register Field Descriptions (Continued)

Field	Description	Settings
SCL5 10–8	**Scale After Stage 5** The number of bits which have been scaled down(shift right) from all inputs after stage 5	Supported values: 0x0–0x4.
— 7–4	Reserved	
OVA_SCL 3–0	**Overall Scaling**. The amount of adaptive scaling that has been applied on the current transform (not including the input scaling, if applied). This field is copied to the FTPE BD status bits ([ADP_OVA_SCL_ST])	Supported values: 0–11.

26.4.4.2.6 FFTPE Saturation Status Register 0 (FFTPESSTR0)

Offset 0x0007E240 (FFTPESSTR0) Access: User read

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SAT0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-98. FFTPE Saturation Status Register 0

This register records the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-126. FFTPE Saturation Status Register 0 Field Descriptions

Field	Description
— 31–28	Reserved
SAT_IN 27–16	**Saturation Due to Input Scaling** The number of samples that were saturated, due to the input scaling.
— 15–12	Reserved
SAT0 11–0	**Saturation After Pre-Multiplication Stage** The number of samples that were saturated, after the pre-multiplication stage.

26.4.4.2.7 FFTPE Saturation Status Register 1 (FFTPESSTR1)

Offset 0x0007E244 (FFTPESSTR1) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SAT1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SAT2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-99. FFTPE Saturation Status Register 1

This register describes the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-127. FFTPE Saturation Status Register 1 Field Descriptions

Field	Description
— 31–28	Reserved
SAT1 27–16	**Saturation After Stage 1** The number of samples that were saturated, after stage 1.
— 15–12	Reserved
SAT2 11–0	**Saturation After Stage 2** The number of samples that were saturated, after stage 2.

26.4.4.2.8 FFTPE Saturation Status Register 2 (FFTPESSTR2)

Offset 0x0007E248 (FFTPESSTR2) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SAT3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SAT4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-100. FFTPE Saturation Status Register 2

This register describes the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-128. FFTPE Saturation Status Register 2 Field Descriptions

Field	Description
— 31–28	Reserved
SAT3 27–16	**Saturation After Stage 3** The number of samples that were saturated, after stage 3.
— 15–12	Reserved
SAT4 11–0	**Saturation After Stage 4** The number of samples that were saturated, after stage 4.

26.4.4.2.9 FFTPE Saturation Status Register 3 (FFTPESSTR3)

Offset 0x0007E24C (FFTPESSTR3) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SAT5											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-101. FFTPE Saturation Status Register 3

This register records the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-129. FFTPE Saturation Status Register 3 Field Descriptions

Field	Description
— 31–28	Reserved
SAT5 27–16	**Saturation After Stage 5** The number of samples that were saturated, after stage 5.
— 15–0	Reserved

26.4.4.3 DFTPE Registers

26.4.4.3.1 DFTPE Data Size Register 0 (DFTPEDSR0)

Offset 0x0009E140 (DFTPEDSR0) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DS0									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DS1									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-102. DFTPE Data Size Register 0

This register records the data size for iFFTs (data size = [<transform length> - <guards size> - 1] / 4), as described by **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92. These registers are relevant only when DFTPE functions as FFTPE (see **Section 26.3.2.2.5**, *Using the DFTPE as FFTPE*).

Table 26-130. DFTPE Data Size Register 0 Field Descriptions

Field	Description
— 31–26	Reserved
DS0 25–17	**Data Size 0** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 128. The value of the DS0 should not exceed the 2≤ DS0 ≤ (128/4-1) values. For more details see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92.
— 16–10	Reserved
DS1 9–1	**Data Size 1** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 256. The value of the DS1 should not exceed the 2≤ DS1 ≤ (256/4-1) values. For more details see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92.
— 0	Reserved

26.4.4.3.2 DFTPE Data Size Register 1 (DFTPEDSR1)

Offset 0x0009E144 (DFTPEDSR1) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DS2									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DS3									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-103. DFTPE Data Size Register 1

This register records the data size for iFFTs (data size = [<transform length> - <guards size> - 1] / 4), as described by **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92. These registers are relevant only when DFTPE functions as FFTPE (see **Section 26.3.2.2.5**, *Using the DFTPE as FFTPE*).

Table 26-131. DFTPE Data Size Register 1 Field Descriptions

Field	Description
— 31–26	Reserved
DS2 25–17	**Data Size 2** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 512. The value of the DS2 should not exceed the 2≤ DS2 ≤ (512/4-1) values. For details see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 16–10	Reserved
DS3 9–1	**Data Size 3** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 1024. The value of the DS3 should not exceed the 2≤ DS3 ≤ (1024/4-1) values. For details see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 0	Reserved

26.4.4.3.3 DFTPE Data Size Register 2 (DFTPEDSR2)

Offset 0x0009E148 (DFTPEDSR2) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	DS4									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 26-104. DFTPE Data Size Register 2

This register records the data size for iFFTs (data size = [<transform length> - <guards size> - 1] / 4), as described by **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT*, on page 26-92. These registers are relevant only when DFTPE functions as FFTPE (see **Section 26.3.2.2.5**, *Using the DFTPE as FFTPE*).

Table 26-132. DFTPE Data Size Register 2 Field Descriptions

Field	Description
— 31–26	Reserved
DS4 25–17	**Data Size 4** Half the size of the Right/Left symbol carrier in an iFFT process with Transform length of 1536. The value of the DS4 should not exceed the 2≤DS4 ≤ (1024/4-1) values. For details see **Section 26.3.2.2.2.2**, *Guard Band Insertion for iFFT,* on page 26-92.
— 16–0	Reserved

26.4.4.4 DFTPE Status Registers

26.4.4.4.1 DFTPE Status Register (DFTPESTR)

Offset 0x0x0009E238 (DFTPESTR) Access: Read

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TASKIDS								0	Res.			0	0	Res.	Res.
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-105. DFTPE Status Register

This register stores the processing status of the DFTPE. When initiating a new transform, this register is overwritten by the DFTPE. When a new transform execution starts, this register is cleared by the DFTPE.

Table 26-133. DFTPE Status Register Field Descriptions

Field	Description
— 31–16	Reserved
TASKIDS 15–8	**Task ID Status** The task ID status of the command that is currently being executed or of the last command executed.
— 7–0	Reserved

26.4.4.4.2 DFTPE Scaling Status Register (DFTPESCLSTR)

Offset 0x0009E23C (DFTPESCLSTR) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	SCL0			0	SCL1			0	SCL2			0	SCL3		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	SCL4			0	SCL5			0	0	0	0	OVA_SCL			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-106. DFTPE Scaling Status Register

This register records the scaling amount applied for each stage during the Adaptive Scaling mode. It also describes the Overall Scaling amount applied during the whole transform processing.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-134. DFTPE Scaling Status Register Field Descriptions

Field	Description	Settings
— 31	Reserved	
SCL0 30–28	**Scaling After Stage 0** The number of bits which have been scaled down(shift right) from all inputs after stage 0	Supported values: 0x0–0x4.
— 27	Reserved	
SCL1 26–24	**Scaling After Stage 1** The number of bits which have been scaled down(shift right) from all inputs after stage 1	Supported values: 0x0–0x4.
— 23	Reserved	
SCL2 22–20	**Scaling After Stage 2** The number of bits which have been scaled down(shift right) from all inputs after stage 2	Supported values: 0x0–0x4.
— 19	Reserved	
SCL3 18–16	**Scaling After Stage 3** The number of bits which have been scaled down(shift right) from all inputs after stage 3	Supported values: 0x0–0x4.
— 15	Reserved	
SCL4 14–12	**Scaling After Stage 4** The number of bits which have been scaled down(shift right) from all inputs after stage 4	Supported values: 0x0–0x4.
— 11	Reserved	

Table 26-134. DFTPE Scaling Status Register Field Descriptions (Continued)

Field	Description	Settings
SCL5 10–8	**Scaling After Stage 5** The number of bits which have been scaled down(shift right) from all inputs after stage 5	Supported values: 0x0–0x4.
— 7–4	Reserved	
OVA_SCL 3–0	**Overall Scaling** The amount of adaptive scaling that has been applied on the current transform (not including the input scaling, if applied). This field is copied to the FTPE BD status bits ([ADP_OVA_SCL_ST]). If the value calculated is more than 15 (can occur in user-defined scaling), the value recorded in this field will be the overall scaling modulo 16.	

26.4.4.4.3 DFTPE Saturation Status Register 0 (DFTPESSTR0)

Offset 0x0009E240 (DFTPESSTR0) Access: User read

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SAT0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-107. DFTPE Saturation Status Register 0

This register records the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-135. DFTPE Saturation Status Register 0 Field Descriptions

Field	Description
— 31–28	Reserved
SAT_IN 27–16	**Saturation Due to Input Scaling** The number of samples that were saturated, due to the input scaling.
— 15–12	Reserved
SAT0 11–0	**Saturation After Stage 0** The number of samples that were saturated, after stage 0.

26.4.4.4.4 DFTPE Saturation Status Register 1 (DFTPESSTR1)

Offset 0x0009E244 (DFTPESSTR1) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SAT1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SAT2											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-108. DFTPE Saturation Status Register 1

This register records the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-136. DFTPE Saturation Status Register 1 Field Descriptions

Field	Description
— 31–28	Reserved
SAT1 27–16	**Saturation After Stage 1** The number of samples that were saturated, after stage 1.
15–12	Reserved
SAT2 11–0	**Saturation After Stage 2** The number of samples that were saturated, after stage 2.

26.4.4.4.5 DFTPE Saturation Status Register 2 (DFTPESSTR2)

Offset 0x0009E248 (DFTPESSTR2) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SAT3											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	SAT4											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-109. DFTPE Saturation Status Register 2

This register records the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-137. DFTPE Saturation Status Register 2 Field Descriptions

Field	Description
— 31–28	Reserved
SAT3 27–16	**Saturation After Stage 3** The number of samples that were saturated, after stage 3.
— 15–12	Reserved
SAT4 11–0	**Saturation After Stage 4** The number of samples that were saturated, after stage 4.

26.4.4.4.6 DFTPE Saturation Status Register 3(DFTPESSTR3)

Offset 0x0009E24C (DFTPESSTR3) Access: User read

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	SAT5											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26-110. DFTPE Saturation Status Register 3

This register records the number of samples in which a saturation event occurred during calculation. There is a field for each transform stage.

Note: Due to internal pipeline implementation of the FTPE, the values of the register are valid only during the output data phase.

Table 26-138. DFTPE Saturation Status Register 3 Field Descriptions

Field	Description
— 31–28	Reserved
SAT5 27–16	**Saturation After Stage 5** The number of samples that were saturated, after stage 5.
— 15–0	Reserved

Security Engine (SEC) 27

The Security Coprocessor version 3.1.0 (SEC) performs computationally intensive security functions including the following:

- Key generation and exchange
- Authentication
- Bulk encryption.

It is optimized to process all the algorithms associated with internet protocol security (IPSec), internet key exchange (IKE), secure sockets layer/transport layer security (SSL/TLS), internet small computer system interface (iSCSI), secure real-time transport protocol (SRTP), the **IEEE** 802.11i™ security standard, worldwide interoperability for microwave access (WiMAX), third generation (G3) A3/5 for global system for mobile communication (GSM) and Enhanced Data Rates for GSM evolution (EDGE), and GEA3 for general packet radio service (GPRS). For applications requiring security protection for sensitive data, the SEC provides encryption/decryption capability without imposing process loading on the device core processors. The SEC includes a controller, four data channels, and eight execution units (EUs) with a shared random number generator (RNGU) that use a common interface to the controller. The EUs perform the specific mathematical manipulations required by protocols used in cryptographic processing.

27.1 Architecture Overview

Since the SEC can act as a master on the internal system bus, it does not impose the data movement bottleneck on the core processing that is normally associated with slave-only processors. In addition, the SEC resides on the peripheral memory map of the processor, and therefore, when an application requires cryptographic functions, it simply creates descriptors for the SEC that define the cryptographic function to perform and the location of the data. The SEC bus-mastering capability permits the core processor to set up a channel with a few short register writes, leaving the SEC to perform reads and writes on system memory to complete the required task.

The security engine includes 8 different execution units (EUs). For EUs for which data flows in and out, each EU has buffer FIFOs of at least 256 bytes. EU types and features include the following:

- Advanced Encryption Standard Unit (AESU)
  - Implements the Rijndael symmetric key cipher per U.S. National Institute of Standards and Technology FIPS 197.
  - Modes providing data confidentiality: ECB, CBC, CCM, Counter, GCM, XTS, CBC-RBP, OFB-128, and CFB-128.
  - Modes providing data authentication: CCM, GCM, CMAC (OMAC1), and XCBC-MAC.
  - 128, 192, 256 bit key lengths (only 128 bit keys in XCBC-MAC)
  - Integrity Check Vector (ICV) checking in CCM, GCM, CMAC (OMAC1), and XCBC-MAC mode
  - XOR operations on 2–6 sources for RAID applications
- ARC Four Execution Unit (AFEU)
  - Implements a stream cipher compatible with the RC4 algorithm
  - 8-bit to 128-bit programmable key
- Cyclic Redundancy Check Unit (CRCU)
  - Implements CRC32C as required for iSCSI header and payload checksums, CRC32 as required for **IEEE** Standard 802 packets, as well as for programmable 32 bit CRC polynomials
  - ICV checking
- Data Encryption Standard Execution Unit (DEU)
  - DES, 3DES
  - Two key (K1, K2, K1) or Three Key (K1, K2, K3)
  - ECB, CBC, CFB-64 and OFB-64 modes for both DES and 3DES
- Kasumi Execution Unit (KEU)
  - Implements cipher and authentication modes F8 and F9 used in 3G, A5/3 for GSM and EDGE, and GEA3 for GPRS

  - 128-bit confidentiality key and 128-bit integrity key
  - ICV checking for F9
- SNOW3G Execution Unit (STEU)
  - Implements cipher and authentication modes UEA2 (F8) and UIA2 (F9)
  - 128-bit confidentiality key and 128-bit integrity key
  - ICV checking for F9
- Message Digest Execution Unit (MDEU)
  - Implements SHA with 160-bit, 224-bit, 256-bit, 384-bit, and 512-bit message digest (as specified by the FIPS 180-2 standard)
  - Implements MD5 with 128-bit message digest (as specified by RFC 1321)
  - Implements HMAC computation with either message digest algorithm (as specified in RFC 2104 and FIPS-198)
  - Implements SSL MAC computation
  - ICV checking
- Public Key Execution Unit (PKEU)
  - RSA and Diffie-Hellman with programmable field size up to 4096 bits
  - Elliptic curve cryptography
    - F_2m and F_p modes
    - Programmable field size up to 1023 bits
  - Run time equalization to protect against timing and power attacks
- Random Number Generator (RNGU). Combines a True Random Number Generator (TRNG) and a deterministic Pseudo-Random Number Generator (PRNG), based on SHA, as described in FIPS 186-2, Appendix 3.1.

In addition to the execution units, the SEC also includes:

- A context switching polychannel, permitting operation of up to four virtual channels, where each channel:
  - Can be configured for any of the core processors
  - Supports a queue of commands (descriptor pointers) to be executed
  - Provides dynamic arbitration for needed crypto-execution units
  - Manages up to two execution units (one ciphering and one hashing), and configures for any required data transfers from one to another
  - Performs flow-control management of buffer FIFOs on the inputs and outputs of execution units
  - Supports scatter/gather of input and output data (where the term data is used loosely, and includes keys, context, ICV values, and so forth), enabling concatenation of multiple segments of memory when reading or writing data
  - Masters data bursts on 32-byte boundaries to optimize bus throughput.
- Master and slave interfaces, with DMA capability

— 32-/36-bit address and 64-bit data
— Up to 333 MHz operation
— Master interface allows pipelined requests
— DMA data blocks can start and end on any byte boundary

27.1.1 Functional Diagram

A functional diagram of the SEC internal architecture is shown in **Figure 27-1**. The controller block is designed to transfer 64-bit sets between the bus and any register inside the SEC.

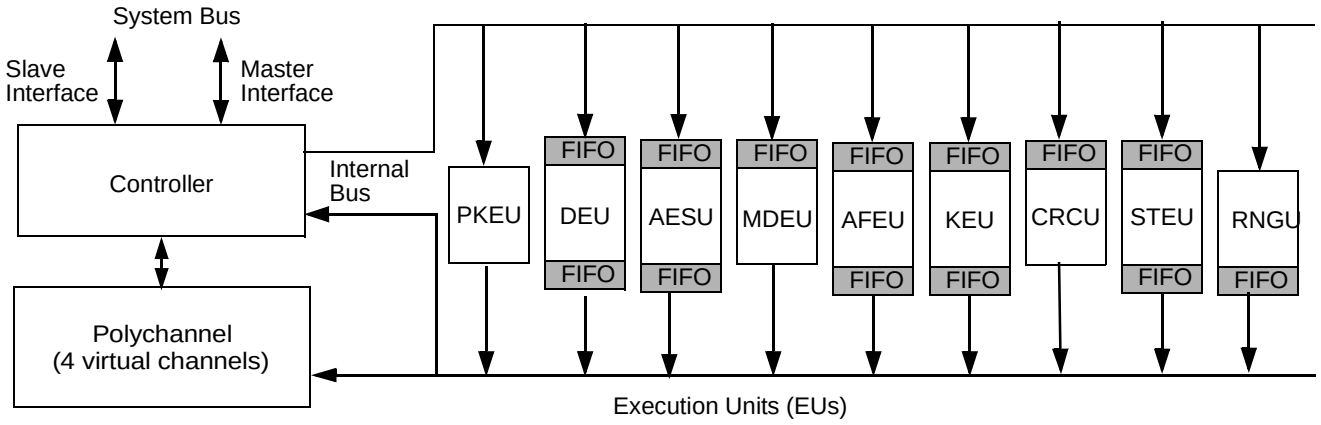


Figure 27-1. SEC Functional Diagram

The SEC interfaces with the system buses through the controller. The Slave Interface permits an external device to perform 32-bit or 64 bit writes on any register or FIFO inside the SEC Some locations permit byte writes. Reads may be of any length. Using the Master Interface, the controller can transfer blocks of 64-bit words between system memory and SEC FIFOs or registers.

A typical SEC operation begins when the core processor writes a descriptor pointer to the Fetch FIFO in one of the four SEC virtual channels. This write operation uses the slave interface (in which the core processor is master and SEC is the slave). Following the write, the channel directs the sequence of operations, using the master interface (for which the SEC is master). The channel uses the descriptor pointer to read the descriptor, then decodes the first 4 bytes of the descriptor to determine the operation to perform and the crypto-execution units needed to perform it. The channel requests the controller to assign the needed crypto-execution units. If necessary, the channel waits for the needed execution units to be free.

Next, the channel requests that the controller transfer the keys, context, and data from locations specified in the rest of the descriptor to the appropriate execution units. The controller satisfies the requests by making requests to the master interface per the programmable priority scheme. Data is fed into the execution units through their registers and input FIFOs. The execution units read from their input FIFOs and write processed data to their output FIFOs and registers. The

channel requests that the controller write data from the output FIFOs and registers back to system memory.

The channel indicates completion by sending a descriptor via an interrupt to the core processor or by a writeback of the descriptor header into core processor memory. For details about this signaling, see **Section 27.1.3**, *Polychannel*.

Upon completion of a descriptor, the channel checks the next entry in its Fetch FIFO, and, if non-zero, requests a burst read of the next descriptor. For most packets, the entire payload is too long to fit in an execution unit input or output FIFO. Therefore, the channel uses a flow control scheme for reading and writing data. The channel directs the controller to read bursts of input as necessary to keep refilling the input FIFO until the entire payload is fetched. Similarly, the channel directs the controller to write bursts of output whenever enough accumulates in the execution unit output FIFO.

The Polychannel processes up to four descriptors concurrently by implementing four virtual channels. Channels arbitrate for use of execution units, and wait if the needed execution unit is currently reserved by another channel. Each channel has its own FIFO of descriptor pointers to fetch and execute, and its own internal storage. The four channels, however, time-share a single control and datapath unit, and hence they are referred to as virtual channels. A programmable priority scheme uses round-robin or weighted priorities among these channels.

27.1.2 Controller

The controller manages the master and slave interfaces to the device system interface and the internal buses that connect all the various SEC modules. It receives service requests from the core processor (via the slave interface) and from the channels and schedules the required data transfers. The system bus interface and access to system memory are critical factors in performance, and the 64-bit master and slave interfaces of the SEC controller enable it to achieve performance unattainable on secondary buses.

27.1.2.1 Controller Operation

The controller can interface with the execution units in two ways:

- Channel-controlled access—A channel can request a particular service from any available execution unit. This is the normal operating mode.
- Core processor-controlled access—The core processor can move data into and out of any execution unit directly through memory-mapped EU registers. Typically, this is only used for debug. Intervention by a core during normal operation may drive an executing EU into an error condition.

27.1.2.1.1 Channel-Controlled Access

This type of access is the normal SEC operating mode. All direct interaction with the EUs is directed by the channel executing the descriptor. The core processor provides the initial descriptor pointer and handle the results when processing is complete, but is not involved in the EU processing. Processing begins when a descriptor pointer is written to the Fetch FIFO of one of the channels. The typical processing step are listed in **Section 27.1.4.1**.

27.1.2.1.2 Core Processor-Controlled Access

All execution units (EUs) are memory-mapped, and can be used entirely through register read/write access. The SEC operates as a slave, and the core processor must write the information normally provided through the descriptor into the appropriate registers and FIFOs of the SEC. This method is more processor intensive and requires a great deal of familiarity with the SEC registers. It is recommended that core processor-controlled access be used only for operations using a single EU and for debug purposes.

27.1.2.2 Descriptors and Link Tables

Since the SEC was designed for easy use and integration with existing systems and software, it uses descriptors to access all cryptographic functions. A descriptor specifies the cryptographic function to perform and contains pointers to all necessary input and output data locations. Some descriptor types perform multiple functions as required by specific protocols.

Each descriptor contains eight 64-bit/8-byte sets, consisting of the following:

- One 64-bit header. The header describes the required services and encodes information that indicates the EUs to use and the modes to set. It also indicates whether notification should be sent to the core processor when the descriptor operation is complete.
- Seven 64-bit pointers used to locate input or output data. Each pointer can point either directly to the data or to a link table that lists a set of data segments to concatenate. Link tables are used by the descriptors to allow concatenation of data parcels as required.

A sample descriptor is described in **Table 27-1**.

Table 27-1. Example Descriptor

Field Name	Value	Description
Header	0x20531E0800000000	Example header for IPsec ESP outbound using DES and MD-5
Length0 Extent0 Pointer0	16 0 (32 or 36-bit pointer)	Number of bytes in authenticate key Unused Pointer to authentication key
Length1 Extent1 Pointer1	16 0 (32 or 36-bit pointer)	Number of bytes in authentication-only data Unused Pointer to authentication-only data
Length2 Extent2 Pointer2	8 0 (32 or 36-bit pointer)	Length of input context (initialization vector—IV) Unused Pointer to input context

Table 27-1. Example Descriptor (Continued)

Field Name	Value	Description
Length3 Extent3 Pointer3	8 0 (32 or 36-bit pointer)	Number of bytes in cipher key Unused Pointer to cipher key
Length4 Extent4 Pointer4	1500 0 (32 or 36-bit pointer)	Number of bytes of data to be ciphered Unused Pointer to input data to perform ciphering upon
Length5 Extent5 Pointer5	1500 12 (32 or 36-bit pointer)	Number of bytes of data after ciphering Number of bytes in authentication result (ICV) Pointer to location where cipher output is to be written, followed by ICV
Length6 Extent6 Pointer6	8 0 (32 or 36-bit pointer)	Length of output Context (IV) Unused Pointer to location where altered Context is to be written

Note: For details, see **Section 27.7.1.1**, *Descriptor Structure*, on page 27-99.

27.1.3 Polychannel

The Polychannel implements four independent virtual channels for processing descriptors. The polychannel is a bridge between the channel operations and the controller and provides direct links to the controller and the EUs. It serves the following functions:

- Sends requests for block transfers received from the EUs to the controller to perform transfers among system memory, the EUs, and the channels.
- Configures the EUs before message processing begins.
- Maintains context when switching tasks.
- Notifies the controller when specified events occur.

The polychannel uses four counters to monitor processing events:

- Fetch FIFO Enqueue Count (FFEC).
- Descriptor Finished Count (DFC).
- Data Bytes In Count (DBIC).
- Data Bytes Out Count (DBOC).

During processing of a channel, the registers increment for each specified event type occurrence. When one of these counters rolls over (changes from maximum to 0), a corresponding bit is set in the Controller Interrupt Status Register if the event interrupt is enabled. This allows the user or system software to preset a value in the counter before processing starts, and then be notified when the preset threshold is reached.

27.1.4 Virtual Channels

Each channel contains the following addressable structures:

- *Fetch FIFO*. Holds a queue of pointers to descriptors waiting to be processed.
- *Configuration register*. Allows the user a number of options for SEC event signalling.
- *Status register*. Indicates the last unfulfilled bus request.
- *Descriptor buffer memory*. Stores the active descriptor and other temporary data.

27.1.4.1 Channel Processing

Whenever a channel is idle and its Fetch FIFO is non-empty, the channel reads the next descriptor pointer from the Fetch FIFO, unless the processing engine operation is stopped (which may require reinitialization or reset of the EU). Using this pointer, the channel fetches the descriptor and places it in its descriptor buffer. To process this descriptor, the channel directs execution of the following steps.

1. Analyze the descriptor header to determine the cryptographic services required, and request use of the appropriate EU(s) from the controller.
2. If required EUs are reserved by another channel, wait for the EU(s) to be available. If available, reserve the EU(s).
3. Configure the appropriate mode bits in the EU(s) for the required EU function.
4. Fetch parcels (up to 64K – 1 bytes long) from system memory using pointers from the descriptor buffer, and place them in either an EU input FIFO or EU registers (as appropriate). The term parcel refers here to any input or output of a cryptographic process, such as a key, hash result, input context, output context, or text-data. Context refers to either an initialization vector (IV) or other internal EU state that can be read out or loaded in. Text-data refers to the plaintext or ciphertext on which to operate. Each parcel transfer may involve using link tables to gather input data that has been split into multiple segments in system memory.
5. If the data size is greater than EU FIFO size, continue fetching input data and writing output data to memory, as required.
6. After writing the last input data to each EU input FIFO, write to the End of Message Register in the EU.
7. Wait for EU(s) to complete processing of text-data.
8. Upon completion, unload final results from the output FIFOs and Context Registers and write them to external memory using pointers in the descriptor buffer. This may involve using link tables to scatter output data into multiple segments in system memory.
9. Reset and release the EUs.

10. If done notification is enabled, perform this notification to the core processor to indicate descriptor completion.

27.1.4.2 Channel Completion

The channel indicates completion by sending a descriptor via an interrupt to the core processor or by a writeback to the descriptor header. In the case of a writeback, the value written back is identical to the header that was read, except for a DONE field, which is set to all 1s. The user performs this notification signalling at the end of every descriptor or at the end of selected descriptors.

27.1.4.3 Integrity Check Value (ICV) Generation and Checking

An EU operation can include generating an ICV and then comparing it against a received ICV. The result of the ICV checking can be sent to the core processor via interrupt or by a writeback of the descriptor header (but not by both methods). If both are enabled, the occurrence of an error interrupt prevents the writeback from occurring.

In the case of a writeback, the user can opt to do it at the end of every descriptor or only at the end of descriptors that call for ICV comparison. In the case of an error condition in a channel or its reserved EUs, the channel issues an interrupt to the core processor. The channel can be configured either to abort the current descriptor and proceed to the next one, or to halt and wait for core processor intervention.

Note: For details on configuring signaling, see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-196. For detail on the writeback fields, see **Section 27.7.1.2**, *Descriptor Header*, on page 27-100.

27.1.4.4 Encryption and Hashing

Many security protocols involve both encryption and hashing of packet payloads. To accomplish this without requiring two passes through the data, channels can configure data flows through two EUs. In such cases, one EU is designated as the primary EU, and the other as the secondary EU. The primary EU receives its data from memory via the controller, and the secondary EU receives its data by snooping the SEC buses.

27.1.4.5 Snooping

There are two types of snooping.

- Input data can be fed to the primary EU and the same input data is snooped by the secondary EU. This is called in-snooping.
- Output data from the primary EU can be snooped by the secondary EU. This is called out-snooping.

Note: In the SEC, only the MDEU and the CRCU can be selected as the secondary EU. For details, see **Section 27.3**, *Polychannel*, on page 27-15.

27.1.5 Common EU Interface

The controller and the channels use a standardized common interface to the EUs. As shown in **Figure 27-2**, the main access to the EUs is through common input and output buses. EU FIFOs and registers are memory-mapped on these buses with 4 Kbytes allocated for each EU. Within each EU memory segment, standard addresses are used for common register types. In addition to this bus interface, each EU supplies three interrupt signals to the controller. Two of these (done interrupt and error interrupt) are routed by the controller to the channel currently using that EU.

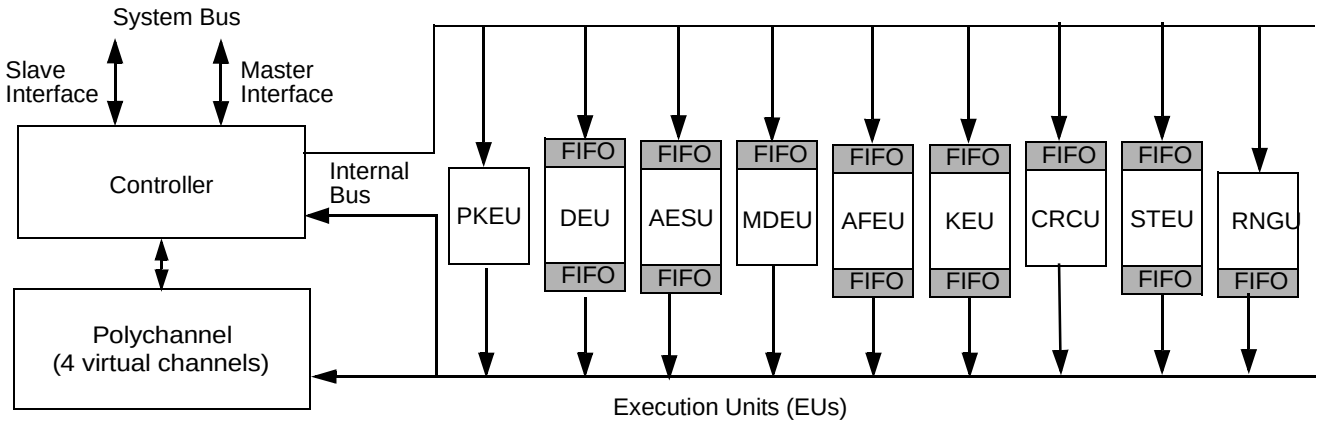


Figure 27-2. SEC Functional Diagram

The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and **IEEE** Std. 802.11i, WiMAX, 3G, A5/3 for GSM and EDGE, and GEA for GPRS, processing and can work together to perform high-level cryptographic tasks. Each EU is described in detail in **Section 27.6**, *Execution Units*, on page 27-24.

27.2 SEC Controller

The controller within the SEC is responsible for mastering bus transfers on behalf of the channels. The controller interfaces to the core processors via the master and slave bus interfaces and to the channels and EUs via internal buses. All transfers between the core processors and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Accept and execute commands from the slave bus to read or write memory-mapped locations (up to 64 bits) anywhere in the SEC.
- Accept and execute requests from the polychannel to transfer blocks of bytes among system memory, EUs, and the channels.
- Realign read and write data using the proper byte alignment.
- Monitor interrupts from channels and pass them to a core processor.

27.2.1 Bus Transfers

The controller is involved in transfers on the system bus and the internal bus. The system bus actually refers to two buses:

- Slave bus for operating SEC as a slave
- Master bus for transactions with SEC as the master

The internal bus is a private 64-bit slave bus with the controller block as the sole master. All accesses to SEC from the system bus go through the controller. The controller also directs transfers over the internal bus.

For core-controlled access, the core uses the external slave bus to access the controller as a slave, and the controller relays the read or write accesses to the proper block over the internal bus. When a write command is received from the system bus, the controller takes the data and sends it to whichever internal location is indicated by the address. For a read, the controller goes to the internal location, fetches the requested data from the specified address (if allowed), and returns it over the system bus.

For channel-controlled access, the channels make requests to the controller to perform data transfers. The channel specifies data lengths and addresses for the internal and system buses. The controller can queue up to four requests. The controller dequeues requests and performs the required transfer. Most transfers involve not only the internal bus, but also the external master bus with the controller as bus master. Here are examples of the various types of bus-master transfers:

- Obtaining a descriptor:
  - channel makes request
  - controller arbitrates for use of the system bus and performs read from external memory
  - controller sends descriptor to channel over the internal bus
- Transferring data length parameter from a channel to the EU
  - channel makes request
  - controller transfers data from channel to EU over the internal bus
- Obtaining input data from external memory for input to an EU
  - channel makes request
  - controller arbitrates for use of the system bus and performs read from external memory
  - controller sends data to EU over the internal bus. If insnooping, data is sent to two EUs
- Writing output data from an EU back to external memory
  - channel makes request
  - controller begins reading data from the EU into a controller FIFO. If outsnooping, the same data is also read by another EU. Meanwhile, the controller arbitrates for use of the system bus.
  - controller performs write to external memory

- Status writeback
  - channel makes request
  - controller reads value from channel and arbitrates for use of the system bus
  - controller performs write to external memory, overwriting the writeback field of the descriptor's header

27.2.1.1 System Bus Master Read

The following list defines the sequence of events for a system bus read with the controller as master:

1. Channel asserts bus read request to the controller
2. Channel furnishes external read address, internal write address, and transfer length
3. Controller asserts request to the system bus through the Magenta master interface
4. Controller waits for system bus read to begin
5. When bus read begins, controller receives data from the master interface and performs a write to the appropriate internal address supplied by the channel. Data may be realigned byte-wise by the controller if either:
   - the external read address was not on an 8-byte boundary, or
   - the internal write address was not on an 8-byte boundary.
6. Transfer continues until the bus read is completed and the controller has written all data to the appropriate internal address. The master interface will continue making bus requests until the full data length has been read.

When the SEC performs a transaction as master, it is possible for the intended slave to terminate the transfer due to an error. The SEC transaction requests are posted to a target queue, after which the core must take responsibility for completing the transaction or signalling error. An error in an SEC initiated transaction is also reported by the SEC via the Channel Interrupt Status Register. The core processor can determine which channel generated the interrupt by checking the ISR for the channel ERROR bit.

27.2.1.2 System Bus Master Write

The following list is the sequence of events for a system bus write with the controller as master:

1. Channel asserts its bus write request to the controller.
2. Channel furnishes internal read address, external write address, and transfer length.
3. Controller performs a read from the appropriate internal address supplied by the channel, loads the write data into its FIFO, asserts a request to the system bus through the master bus interface, and waits for the system bus to become available.

4. When the system bus becomes available, the controller writes data from its FIFO to the master interface.

27.2.2 Controller Interrupts

The controller produces a single interrupt to each core, either a primary interrupt or a secondary interrupt. Only the Channel Error, Channel Done and Channel Done Overflow status bits can cause the secondary interrupt to assert. All other interrupt types assert the primary interrupt.

27.2.2.1 Controller Primary Interrupt

All interrupt outputs from other SEC blocks are fed to the controller as interrupt conditions. In addition, the controller itself detects some interrupt conditions. The controller maintains an Interrupt Status Register (ISR) with bits corresponding to all of these possible interrupt conditions. If an interrupt condition occurs and the corresponding bit of the Interrupt Enable Register (IER) is set, then the associated Interrupt Status Register bit is set, indicating the presence of a pending interrupt. When any bits are set in the Interrupt Status Register, the controller asserts its primary interrupt output line to the core processor. A primary interrupt generates an IRQ index vector of 201 in the EPIC modules in each core subsystem.

The SEC also allows configuration of specific channel interrupt conditions (specifically, Channel Error, Channel Done and Channel Done Overflow) as secondary interrupts by remapping the channel status register to its alternative (secondary) address using one of the RCA bits in the Master Control Register (MCR) (see **Section 27.7.4.1**, *Master Control Register (MCR)*, on page 27-177). If one of these channel errors occurs in this configuration, the controller generates a secondary interrupt to the cores (using a separate interrupt line). This generates an IRQ index vector of 202 in the EPIC modules in each core subsystem.

To handle an interrupt, the core processor must read the Interrupt Status Register to determine the source. It may then need to do further reads of interrupt status registers of other blocks to get more detailed information about the cause. In the case of a secondary interrupt, the core would only have to read the specific channel status register, but at the secondary address.

In some cases, the core processor may need to take action to clear the root cause of the interrupt. After that, the core processor can clear the desired bit of the Interrupt Status Register by writing a 1 to the corresponding bit of the Interrupt Clear Register (ICR). If the cause of the interrupt condition is not cleared, or if there is some other interrupt condition from the same source, then the Interrupt Status Register bit will clear for a cycle and go high again, and the interrupt output line to the core processor remains high. If the Interrupt Status Register bit is successfully cleared and no other interrupt conditions are present, the controller de-asserts its interrupt output. If any interrupts are still pending in the Interrupt Status Register, the interrupt output remains asserted. This process is valid for both the primary and secondary interrupt lines independently.

Note that EU interrupt conditions may be blocked at two different levels. There is an Interrupt Mask register in each EU that can block particular interrupt conditions before they reach the EU interrupt status register. In addition, interrupts from EUs can be individually blocked by bits of the controller Interrupt Enable Register before they reach the controller Interrupt Status Register. For normal operation, interrupts from EUs are typically disabled in the controller's Interrupt Enable Register, but they still reach the channel, and the channel produces Done or Error interrupts to the core processor as needed. Interrupt conditions from the channels and controller can only be blocked through the Controller Interrupt Enable Register.

A channel can generate frequent interrupts, especially if it is configured to interrupt at the completion of each descriptor. To make sure that the core processor receives the right number of interrupts, each channel Done interrupt has a special queuing feature. If multiple Channel Done interrupts are generated before the first is cleared, then the additional interrupts are counted by the controller. Each time the core processor clears a channel interrupt, the count is decremented. If the core processor clears the channel interrupt and the count reaches zero, the Channel Done interrupt is deasserted. If the count does not reach zero, the controller deasserts the interrupt for one cycle and then re-asserts it again.

Up to 15 interrupts can be queued for each channel. If the count of queued interrupts for any channel exceeds 15, the channel Done Overflow bit is set (see **Section 27.7.4.6**).

27.2.2.2 Controller Secondary Interrupt

Whenever the Channel Done, Channel Error or Channel Done Overflow bits are set in the Interrupt Status Register and the channel has been remapped to the alternative address by one of the RCA bits in the Master Control Register (MCR) (see **Section 27.7.4.1**, *Master Control Register (MCR)*, on page 27-177), the controller asserts the secondary interrupt output line to the secondary core processor. No other Interrupt Status Register status bits can cause the secondary interrupt to assert.

27.2.3 Controller Registers

The controller uses the following registers and structures:

- Master Control Register (MCR). This register allows the user to impose a real-time priority level on the internal arbiter for the controller bus access, to generate a software reset using a write to this register, and set the EU and bus priority counts for Channels 3 and 4. For details on this register and its programming model, see **Section 27.7.4.1**, *Master Control Register (MCR)*, on page 27-177.
- Controller Identification Register (CIR). This register provides an ID for the SEC used by software to verify the supported security revision levels. It is a duplicate of the information in the Controller IP Block Revision. For details on this register, see **Section 27.7.4.2**, *Controller Identification Register (CIDR)*, on page 27-180.

- Controller IP Block Revision Register (CIPBRR). This register contains a 64-bit value that identifies the version of the SEC 3.1 protocol supported by this device. For details on this register, see **Section 27.7.4.3**, *Controller IP Block Revision Register (CIPBRR)*, on page 27-180.
- Controller EU Assignment Status Register (CEUASR). This register records the assignment for each EU to a channel. If the EU is assigned a channel, it becomes inaccessible to any other channel. For details on this register, see **Section 27.7.4.4**, *EU Assignment Status (EUASR)*, on page 27-181.
- Controller Interrupt Enable Register (CIER). The CIER allows the user to enable or disable interrupt generation by specific sources. After reset, all sources are disabled. When a source is enabled by setting the corresponding bit in the CIER, it can set a bit in the Interrupt Status Register (ISR) which generates an interrupt to the core processor. For normal operation, enable all the channel interrupts and disable all the EU-specific interrupts. The channels generate the appropriate interrupts to the core processor. For details on this register, see **Section 27.7.4.5**, *Controller Interrupt Enable Register (CIER)*, on page 27-182.
- Controller Interrupt Status Register (CISR). The ISR contains fields representing all possible sources of interrupts. The Interrupt Status Register is cleared either by a reset, or by writing the appropriate bits active in the Interrupt Clear Register. For details on this register, see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186.
- Controller Interrupt Clear Register (CICR). The CICR provides a means clear the CISR. Writing a 1 to a bit in the CICR clears the corresponding bit in the ISR. If no other interrupt is pending, it also deasserts the interrupt output pin $\overline{\text{IRQ}}$. If the input source to the ISR remains active, the appropriate ISR bit sets the $\overline{\text{IRQ}}$ is reasserted shortly thereafter. The ICR bit clears automatically clear one cycle after it is written. For details on this register, see **Section 27.7.4.7**, *Controller Interrupt Clear Register (CICR)*, on page 27-189.

27.3 Polychannel

The polychannel is the main control unit in the SEC. It implements four independent channels for processing descriptors. The polychannel is a bridge between the channel operations and the controller and provides direct links to the controller and the EUs. It serves the following functions:

- Sends requests for block transfers received from the EUs to the controller to perform transfers among system memory, the EUs, and the channels.
- Configures the EUs before message processing begins.
- Maintains context when switching tasks.
- Notifies the controller when specified events occur.

The polychannel uses four counters to monitor processing events:

- Fetch FIFO Enqueue Count (FFEC).
- Descriptor Finished Count (DFC).
- Data Bytes In Count (DBIC).
- Data Bytes Out Count (DBOC).

During processing of a channel, the registers increment for each specified event type occurrence. When one of these counters rolls over (changes from maximum to 0), a corresponding bit is set in the Controller Interrupt Status Register if the event interrupt is enabled. This allows the user or system software to preset a value in the counter before processing starts, and then be notified when the preset threshold is reached.

27.4 Channels

The individual channels in the polychannel manage the execution of each cryptographic task, making use of one or more of the SEC execution units (EUs). Control information and data pointers for a given task are stored in the form of a descriptor (see Section 27.7.1, **Descriptors and Link Tables**) in system memory or in the channel itself. A descriptor determines which EUs are used, how they are configured, where to fetch needed data, and where to store the results.

27.4.1 Channel Operation

To invoke cryptographic tasks, the core processor constructs a descriptor, selects a channel, and writes a pointer to the descriptor into the selected channel Fetch FIFO. The Fetch FIFO can store up to 24 pointers. Operations performed by channels include the following (not necessarily in this order):

- If the channel is idle and its Fetch FIFO is non-empty, it reads the next descriptor pointer from the Fetch FIFO, and uses the pointer to read the descriptor into the channel descriptor buffer.
- Requests from the controller the assignment of one or more EUs for the exclusive use of the channel. When necessary, configures the secondary EU to snoop input or output data intended for the primary EU.
- Upon notification of completion of the EU reset sequence, initializes Mode Registers in the assigned EU.
- Initializes EUs and writes to EU registers (such as key size and text-data size).
- Transfers data parcels (up to 32 Kbytes) from system memory into the assigned EU input registers and FIFOs. This can involve using link tables to gather input data that is split into multiple segments stored in various locations in system memory. For the RAID-XOR descriptor type, the channel rotates among three data sources, fetching 32 bytes from each source.

- Transfers data parcels (up to 32 Kbytes) from assigned EU output registers and FIFOs to system memory space. This can involve using link tables to scatter output data into multiple segments which are stored in various locations of system memory.
- Initialize the End_of_Message Register (where applicable) in the assigned EU upon completion of last EU write indicated by the descriptor. The channel waits for a indication from the EU that processing of input text-data is complete before proceeding with further activity after writing end_of_message.
- Resets assigned EU(s).
- Releases assigned EU(s).
- When a descriptor is completely processed, provides feedback to the core processor, in the form of an interrupt and/or descriptor header write-back to system memory.
- When descriptor processing is halted due to an error, provides feedback to the core processor via an interrupt.
- The channel waits indefinitely for the controller to complete a requested activity before continuing to the next step of descriptor processing.
- The channel can generate two types of done notification signals when it completes operation on a descriptor—an interrupt and/or a writeback of the descriptor header.
- If selected, the done notification is performed at the end of processing either for every descriptor or selected descriptors.
- If enabled, the channel can also write back status information from the EU(s) involved in processing the descriptor.

Note: The done and status writebacks are not performed if the EU(s) generate any error during processing. The detected error can include, but is not limited to, a failing, unmasked ICV Check in an EU.

27.4.2 Arbitration Among Channels

All channels share a set of common resources, namely, use of EUs, use of the controller to perform data transfers, and use of the polychannel itself to implement channel activity. This section discusses the arbitration mechanisms for these shared facilities.

27.4.2.1 Arbitration for Use of the Polychannel

As previously mentioned, channels are implemented by time-sharing a common datapath and state machine in the polychannel. Each channel’s use of the polychannel is usually brief; a channel typically makes a single request to the controller for a data transfer, computes values for the next transfer, and then goes to sleep, waiting for data transfers to occur or for some EU to process more data.

When the executing channel goes to sleep, it leaves a wakeup condition indicating what event should wake up that channel again. When the wakeup condition is satisfied, the channel is ready

to resume executing. Next time the polychannel becomes free, that channel must arbitrate with any other channels that are also ready to execute.

With the current designs of the controller and magenta-to-copper gasket, there is a delay penalty whenever they change direction from write to read. To maximize data transfer efficiency, it is best to group memory read requests together and memory write requests together. When choosing which channel to execute next, the arbitrator first selects from channels waiting to do writes. When there are no more waiting to write, it begins selecting those waiting to read. It then switches back to handling writes, and so forth.

Within the read or write category, selection of the next channel to execute can be either round-robin or a weighted priority-based scheme, depending on the values of CHN3_BUS_PR_CNT and CHN4_BUS_PR_CNT in the Master Control Register (see **Section 27.7.4.1**). For more information, see **Section 27.4.2.4**.

27.4.2.2 Arbitration for Use of the Controller

No arbitration for use of the controller is necessary. Because channels execute one at a time, a channel experiences no contention when sending a request to the controller. In effect, when a channel wins arbitration for use of the polychannel, it wins use of the controller as well.

27.4.2.3 Arbitration for Use of Execution Units

While one channel has a particular EU reserved, it is possible for multiple other channels to request use of that same EU. Arbitration is necessary to determine which channel will get use of the EU next. To accomplish this, there is an arbiter for each type of execution unit.

EU arbitration can be either round-robin or a weighted priority-based scheme, depending on the values of CHN3_EU_PR_CNT and CHN4_EU_PR_CNT in the Master Control Register (see **Section 27.7.4.1**). For more information, see **Section 27.4.2.4**.

If a channel needs two EUs, a primary and a secondary, it requests them one at a time. Sometimes a channel will reserve one EU and then have to wait for some other channel(s) to finish before obtaining the second requested EU. Though such waiting may occur, the requests are always eventually satisfied. Deadlock is avoided through the following design rules:

1. The channel always requests the secondary EU first.
2. In cases where both a primary and secondary are used, the choices for primaries and secondaries are distinct sets. Primaries are AESU, AFEU, DES, and KFEU, and the secondaries are MDEU and CRCU.

Since primaries and secondaries are distinct sets, and primary and secondary requests are strictly ordered, no deadlock is possible.

27.4.2.4 Arbitration Algorithms

This section applies to both arbitration for use of the polychannel, and arbitration for use of execution units. Control fields for both are in the Master Control Register (see **Section 27.7.4.1**):

- CHN3_BUS_PR_CNT and CHN4_BUS_PR_CNT control polychannel arbitration
- CHN3_EU_PR_CNT and CHN4_EU_PR_CNT control EU arbitration

This section refers to generic control fields CHN3_xx_PR_CNT and CHN4_xx_PR_CNT, where the “xx” refers to either “BUS” or “EU”.

If both CHN3_xx_PR_CNT and CHN4_xx_PR_CNT are zero (the default), arbitration is round-robin, described in Section 27.4.2.4.1. If they are set to non-zero values, the arbiter implements a weighted priority scheme, described in Section 27.4.2.4.2.

The SEC does not dynamically adjust its own transaction priorities. System software, however, can adjust SEC transaction priority in realtime, with the change in priority taking effect immediately.

27.4.2.4.1 Round-Robin Arbitration

In round-robin arbitration, requesting channels are granted access in rotating numerical order: 1, 2, 3, 4, 1, 2, and so on.

27.4.2.4.2 Priority Arbitration

When arbitrating on the priority scheme, the priority is as follows:

- Channel 1—Highest priority
- Channel 2—Second highest priority, unless CHN3_xx_PR_CNT or CHN4_xx_PR_CNT has expired
- Channel 3—Third priority, unless CHN4_xx_PR_CNT expired
- Channel 4—Lowest priority, until CHN4_xx_PR_CNT expired

Initially, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, a weight-based scheme is used. When channel 3 has lost arbitration the number of times specified in CHN3_xx_PR_CNT, channel 3 replaces channel 2 as the second-highest priority in the next round of arbitration. Likewise, when channel 4 has lost arbitration the number of times specified in CHN4_xx_PR_CNT, channel 4 replaces channel 2 as the second-highest priority in the next round of arbitration. Channel 1 always has the highest priority, but it cannot make back to back requests, so the second highest priority channel will win arbitration either immediately, or after one win from channel 1.

27.4.3 Channel Registers and Structures

Each channel has its own set of registers and structures used for configuration, control, and data manipulation including the following:

- Channel Configuration Register (CCR). Configures the basic channel operation including burst size, extended addressing, and notification type. It also provides control functions such as continuing the operation, resetting the channel, writing back status and DONE, and using an Integrity Check Value (ICV). For details, see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-196.
- Channel Pointer Register (CSR). Contains status fields and counters to report the current status of descriptor processing including the G_STATE (gather) and S_STATE (scatter) fields that report gather and scatter state machine status. For details, see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199.
- Current Descriptor Pointer Register (CDPR). Contains the address for the currently processing descriptor. For details, see **Section 27.7.6.3**, *Current Descriptor Pointer Register (CDPR)*, on page 27-204.
- Channel Fetch FIFO (FF). Each channel contains a Fetch FIFO to store a queue of pointers to descriptors to process. In a typical operation, the core processor creates a descriptor in memory containing all relevant mode and location information for the SEC and then launches the descriptor by writing its address to the SEC Fetch FIFO. The Fetch FIFO can hold up to 24 descriptor pointers at a time. When the channel reaches the end of the current descriptor, the next location in the Fetch FIFO is read to launch the next descriptor. The Fetch Address is written into the FIFO only if the write includes the least significant byte (bits 7–0). If the Extend Address Enable (EAE) bit is set, then the Extended Fetch Address must be written before or concurrently with the Fetch Address. Specifying a fetch address of 0 causes the channel to generate an error and stop. For details, see **Section 27.7.6.4**, *Channel Fetch FIFO (CFF)*, on page 27-205.
- Channel Descriptor Buffer (DB). As with any descriptor used by the SEC, the DB consists of an 8-byte header and seven 8-byte pointers with lengths. This structure is read-only because the descriptor is always fetched from system memory. For details, see **Section 27.7.6.5**, *Channel Descriptor Buffer (DB)*, on page 27-206.

27.4.4 Channel Interrupts

The channel can assert done and error interrupts to the controller. The status of the registered channel interrupts is available in the Controller Interrupt Status Register (CISR); see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186 for details. The channel does not have an internal interrupt mask, but the controller can be programmed to block channel interrupts via its Interrupt Enable Register (see **Section 27.7.4.5**, *Controller Interrupt Enable Register (CIER)*, on page 27-182 for details).

27.4.4.1 Channel Done Interrupt

Whether and when a channel done interrupt is generated depends on the setting of the Channel Configuration Register NT and CDIE bits (see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-196). Assuming the CDIE (Channel Done Interrupt Enable) is set, the channel generates an interrupt event after every successfully completed descriptor (Notification Type set to Global), or after each successfully completed descriptor with the DN (Done Notification) bit set in the header of the descriptor. If the EU(s) signal any error during processing, the channel done interrupt is not generated. Even if multiple channel done interrupt events are generated by a channel before the first can be cleared by the core processor, the interrupt events are not lost. The controller queues channel done interrupts from each channel (see **Section 27.2.2**, *Controller Interrupts*, on page 27-13).

27.4.4.2 Channel Error Interrupt

The Channel Error Interrupt is generated when an error condition occurs during descriptor processing. The channel error interrupt is asserted as soon as the error condition is detected. The type of error condition is reflected in the ERROR field of the Channel Pointer Register (CSR). Refer to **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199 for a complete listing of error types.

27.4.4.3 Channel Reset

Channel reset is asserted when the core processor sets the RESET bit in the Channel Configuration Register (CCR). The effect of software reset on the channel varies according to what the channel is doing when the bit is set:

- If the RESET bit is set while the channel is requesting an EU assignment from the controller, the channel canceled its request by asserting the release output signals. The channel then resets all of its registers, clears the RESET bit, and returns the control state machine to the idle state.
- If the RESET bit is set after the channel is dynamically assigned an EU, the channel requests a write from the controller to set the software reset bit of the EU. A write to reset the secondary (MDEU) EU is also requested if one is reserved for snooping. The channel then asserts the appropriate release output signal to notify the controller that the channel has finished with the reserved EU(s). The channel then resets all the registers, clears the RESET bit, and returns the control state machine to the idle state.

27.5 Descriptors and Link Tables

SEC descriptors are designed to support the cryptographic computation of a single packet using a single descriptor. They are conceptually similar to descriptors used by most devices with DMA capability. The descriptors have a fixed length of 64 bytes. A descriptor consists of one header (8 bytes) and seven pointers with lengths (each 8 bytes). See **Section 27.7.1**, *Descriptors and Link Tables*, on page 27-98 for a detailed description of descriptor structure.

The descriptor header indicates which computations to perform and which EUs to use. The descriptor pointers identify input and output data locations. The pointer function capabilities include scatter/gather capability, which means that each pointer in a descriptor can be either a direct pointer to a contiguous parcel of data, or can be a pointer to a link table which is a list of pointers and lengths used to assemble the parcel. **Figure 27-3** shows the layout of a descriptor pointer.

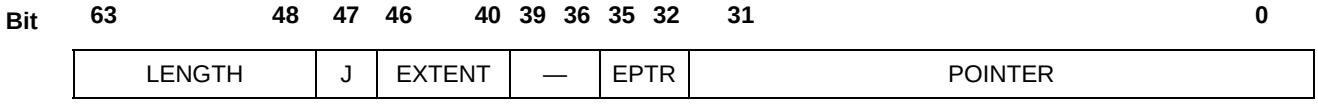


Figure 27-3. Descriptor Pointer Layout

When a link table is used to read input data, this is referred to as a gather operation; when used to write output data, it is referred to as a scatter operation. Although there are only seven pointers in descriptor, a link table identified by the pointer can have any number of 8-byte entries.

There are two kinds of entries, regular entries and next entries. Each regular entry specifies a memory segment by means of a 36-bit starting address (SEGPTR) and a 16-bit length (SEGLEN). A next entry is used at the end of a link table to specify that the list of memory segments is continued in another link table. In a next entry, the N bit is set, the SEGPTR field gives the address of the next link table, and the SEGLEN field must be 0. A chain of link tables may contain any number of link tables. See **Section 27.7.3**, *Link Tables*, on page 27-175 for details on the Link Table programming model. **Figure 27-4** shows the layout of a link table entry.

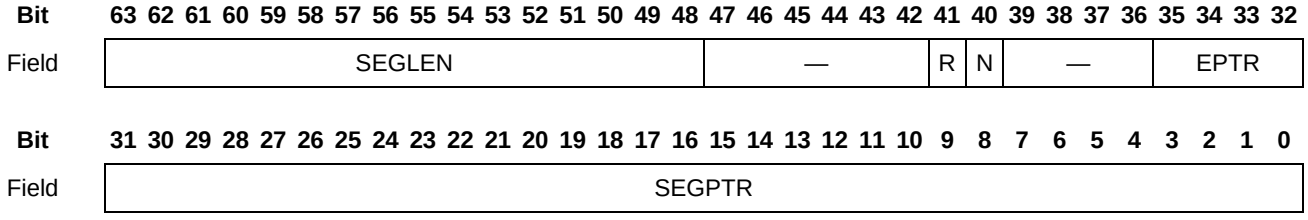


Figure 27-4. Link Table Entry

Whether the list of memory segments is in a single link table or split into several link tables, the last entry in the last link table is a regular entry with the R (return) bit set. The R bit signifies the end of link table operations so that the channel returns to the descriptor for its next pointer (if any).

For any sequence of parcels accessed by a link table or chain of link tables, the combined lengths of the parcels (the sum of their LENGTH and/or EXTENT fields) must equal the combined lengths of the link table memory segments (SEGLEN fields). Otherwise the channel sets the error state in the SGLM bit of the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199).

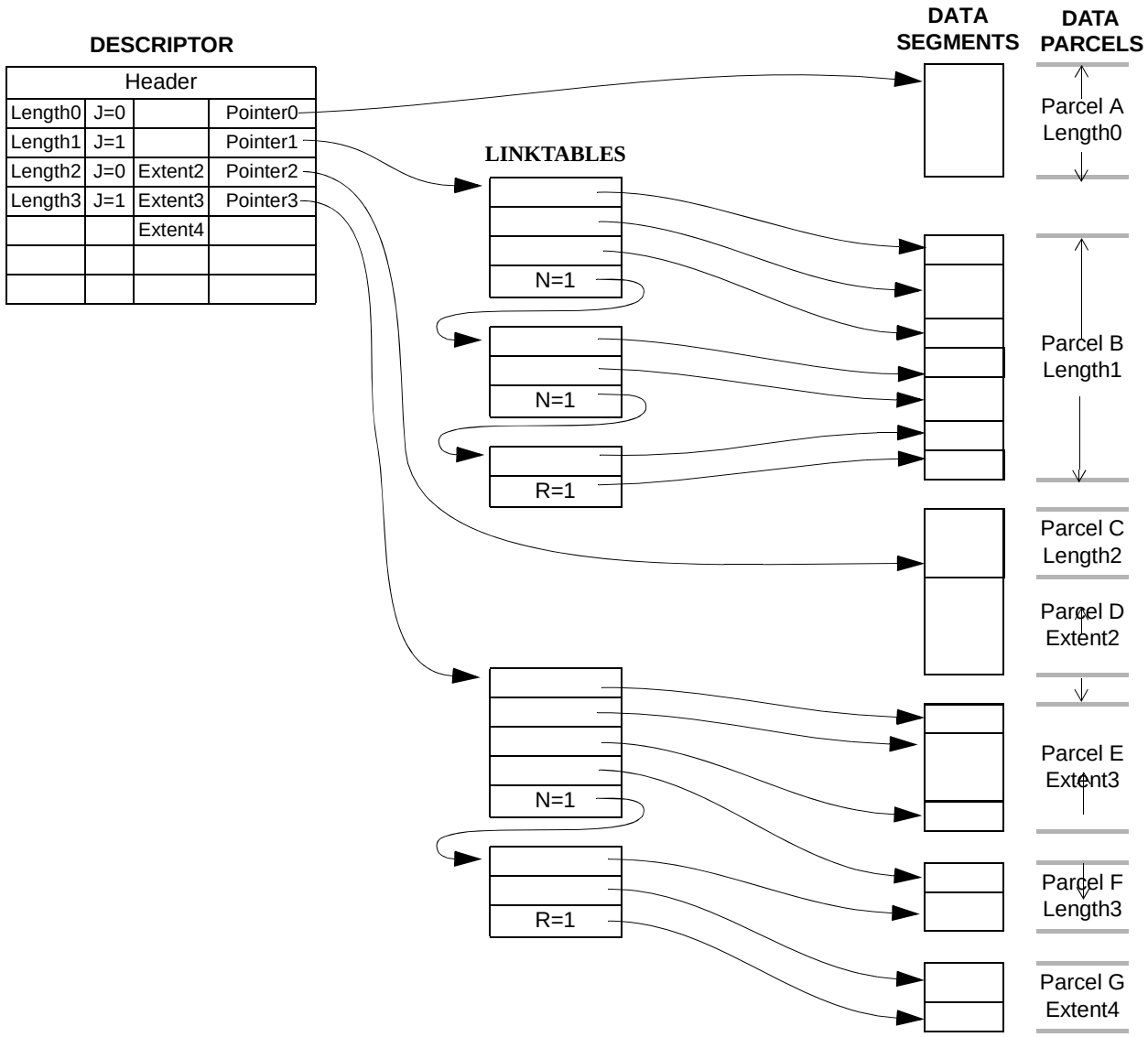


Figure 27-5. Descriptors, Link Tables, and Parcels

Figure 27-5 illustrates various ways that a descriptor can specify parcels:

- The first pointer in the descriptor specifies Parcel A using the simplest method. The pointer identifies the parcel directly through Pointer0 and Length0.
- The second pointer d uses a chain of link tables to specify Parcel B. Since J=1, Pointer1 is used as the address of a link table. The link table specifies several regular entries specifying data segments to be concatenated. The last entry of the link table is a next entry

indicating that the list continues in the next link table. The last entry in the last link table of the chain has the R bit set.

- The last pointers show how one pointer in a descriptor can specify multiple parcels. Pointer2 and Length2 specify Parcel C, then Parcel D follows immediately afterwards, with length specified by Extent2. Pointer3 is used for three parcels (E, F, and G) using link tables.

To demonstrate the use of a link table, assume that the current descriptor type requests the channel to read a parcel using Pointer3 and Extent3 fields, and assume that J3=1. Due to the J3 value, Pointer3 is not used as a data address but instead used as the address of a link table. The channel begins by reading the first four entries starting at Pointer3 into an internal gather table buffer. Using the first entry of the gather table buffer, the channel starts accessing the parcel by reading SEGLEN bytes beginning at SEGPTR. If the required parcel size (Extent3) is greater than this first SegLen, the channel moves on to the next entry of the gather table buffer, and reads SEGLEN bytes starting at SEGPTR. As long as there are more bytes to read in the parcel, the process continues. If the channel gather table buffer is exhausted, the channel reads the next four entries of the link table into its gather table buffer. If a gather table buffer entry is encountered in which the N bit is set, the channel uses the SEGPTR field in that word to find the next link table in the chain.

Now assume that the channel accesses its next parcel using Pointer3 again, this time with length given by Length3. In this case the channel continues to the next line of the link table, and begins reading the memory segment specified there. As before, the channel concatenates memory segments from as many link table entries as necessary to obtain the required number of bytes (Length3).

Similarly, the next parcel is obtained by using Pointer3 yet again, this time with length given by Extent4.

Assume that for the current descriptor type, the Extent4 parcel is the last one to accessed through Pointer3. Then the link table entry that supplies the last memory segment for Extent4 has the R bit set, signifying that this is the last entry in the chain of link tables.

27.6 Execution Units

Execution unit (EU) is the term used for a functional block that performs the mathematical manipulations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and **IEEE** Std. 802.11i, WiMAX, 3G, A5/3 for GSM and EDGE, and GEA for GPRS, processing and can work together to perform high level cryptographic tasks.

The following execution units are used in the SEC:

- Public Key Execution Unit (PKEU)

- Data Encryption Standard Execution Unit (DEU)
- Advanced Encryption Standard Execution Unit (AESU) implementing the Rinjdael symmetric key cipher.
- Message Digest Execution Unit (MDEU)
- ARC Four Execution Unit (AFEU)
- Kasumi (F8/F9) Execution Unit (KEU)
- Cyclic Redundancy Check Unit (CRCU)
- SNOW3G Execution Unit (STEU)
- One private internal Random Number Generator Unit (RNGU)

Working together, the EUs can perform high-level cryptographic tasks, such as the IPSec Encapsulating Security Protocol (ESP) and digital signature. The following sections provide overview of the execution unit operations. Register details are given in the Programming Model at the end of the chapter (**Section 27.7**, *Programming Model*, on page 27-95). In general, direct access to the EU registers is only used for debugging operations.

The mapping each set of EU registers is similar and the location of registers within the memory map uses the same offsets from the specific EU register base address for each of the common EU registers. The EUs and the RNGU all include the following 64-bit registers:

- Mode Register
- Data Size Register
- Reset Control Register
- Status Register
- Interrupt Status Register
- Interrupt Mask Register

In addition to the basic registers, an EU also includes specific registers and data structures to support the individual computational requirements. The register and structure types can include:

- Input FIFOs and/or output FIFOs
- End_of_Message Registers
- Key registers and key memory
- Context registers and context memory and context memory pointers
- Special purpose registers for ABSize, EU-GO, IV, ICV, and data registers

Refer to the individual EU functional descriptions for a list the registers for that EU and to **Section 27.7**, *Programming Model*, on page 27-95 for a detailed description of each register and associated register fields.

27.6.1 Public Key Execution Unit (PKEU)

The public key execution unit (PKEU) is typically operated through channel-controlled access, which means that most reads and writes of PKEU registers are directed by the SEC channels. Driver software performs core processor-controlled register accesses only on a few registers for initial configuration and error handling.

The following subsections include general descriptions of the PKEU registers and structures. **Section 27.7**, *Programming Model*, on page 27-95 provides a detailed description of each register and associated register fields.

27.6.1.1 PKEU Mode Register

This register specifies the internal PKEU routine to execute. The Mode Register is cleared when the PKEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated.

27.6.1.2 PKEU Key Size Register

The register reflects the number of significant bytes to use from PKEU Parameter Memory E in performing modular exponentiation or elliptic curve point multiplication. The range of values for this register when performing either modular exponentiation or elliptic curve point multiplication is from 1 to 512. Specifying a key size outside this range causes a key size error (KSE) in the PKEU Interrupt Status Register.

27.6.1.3 PKEU AB Size Register

This register represents the operand size for the specific operands whenever required. The AB size is in bits even though internally, the PKEU imposes 32-bit alignment. Any data beyond the number specified by the AB Size Register, either in A and B-ram (operands), is ignored. Because no error checking is performed, having an operand size greater than the prime modulus or the field size can result is a wrong result. It is assumed that operands are modulo reduced before being written into the PKEU. Therefore, the AB Size must be less than or equal to the Data Size for a correct result. If the AB Size Register is modified during processing, an error is generated.

27.6.1.4 PKEU Data Size Register

This register specifies, the size of the significant portion of the modulus or irreducible polynomial in bits. Any value written to this register that is a multiple of 32 bits (for example, 128 bits, 160 bits, and so on), is represented internally as the same value (128 bits, 160 bits, respectively). Any value written that is not a multiple of 32 bits (for example, 132 bits, 161 bits, and so on), is represented internally as the next larger 32-bit multiple (160 bits, 196 bits, and so on, respectively). This internal rounding up to the next 32-bit multiple is described for information only. The minimum size valid for all routines to operate properly is 33 bits

(internally 128 bits). The maximum size to operate properly is 2048 bits. A value in bits larger than 2048 results in a data size error.

An illegal data size error is generated as follows:

- All non ECC routines with a data size > 512 generate an illegal data size error.
- All ECC routines with a data size > 128 generate an illegal data size error.
- AB Size = 0 (either intentionally written or by ignoring and not writing at all) generates an illegal size error, except for routines that do not require an A or B operand such as the CLEAR_MEM routine.

27.6.1.5 PKEU Reset Control Register

This register contains three reset options specific to the PKEU.

27.6.1.6 PKEU Status Register

This register contains 6 fields that reflect the state of PKEU internal fields. The PKEU Status Register is read-only. Writing to this location result in an address error being reflected in the PKEU Interrupt Status Register.

27.6.1.7 PKEU Interrupt Status Register

This register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the PKEU Interrupt Mask Register is zero. If the PKEU Interrupt Status Register is non-zero, the PKEU halts and the PKEU error interrupt signal is asserted to the controller (see **Section 27.2.2**, *Controller Interrupts*). In addition, if the PKEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller. If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the PKEU Reset Control Register.

27.6.1.8 PKEU Interrupt Mask Register

This register controls the result of detected errors. For a given error (as defined in Section 27.6.1.7, **PKEU Interrupt Status Register**), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the PKEU Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.1.9 PKEU End_of_Message Register

This register indicate the start of a new computation. Writing to this register causes the PKEU to execute the function requested by the ROUTINE field, per the contents of the parameter memories. This register has no data size, and during the write operation, the core processor data bus is not read. Hence, any data value is accepted.

27.6.1.10 PKEU Parameter Memories.

The PKEU uses four 4096-bit memories to receive and store operands for the arithmetic operations the PKEU performs. In addition, results are stored in one particular parameter memory. Data addressing within these memories is big-endian, that is, the most significant byte is stored in the lowest address.

- PKEU Parameter Memory A. This 4096-bit memory is used typically as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For elliptic curve routines, this memory is segmented into four 1024 bit memories, and is used to specify particular curve parameters and input values.
- PKEU Parameter Memory B. This 4096-bit memory is used typically as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented in to four 1024 bit memories, and is used to specify particular curve parameters and input values, as well as to store result values.
- PKEU Parameter Memory E. This 4096-bit memory is non-segmentable, and stores the exponent for modular exponentiation, or the multiplier k for elliptic curve point multiplication. This memory space is write only; a read of this memory space causes address error to be reflected in the PKEU Interrupt Status Register.
- PKEU Parameter Memory N. This 4096-bit memory is non-segmentable, and stores the modulus for modular arithmetic and F_p elliptic curve routines. For F_2m elliptic curve routines, this memory stores the irreducible polynomial.

27.6.1.11 PKEU Routines

Once a write to End_of_Message Register is issued, the EU checks the Mode Register and if appropriate, the Data Size and Key Size Registers for valid data. If a valid routine is requested (refer to **Section 27.7.7.1, *PKEU Mode Register (PKEUMR), on page 27-207*** for a list of valid routine values), it is performed.

Descriptions of the PKEU routines use the following notational conventions:

- All values are positive
- The modulus and elements of the prime field (F_p) are positive numbers

- The modulus and elements of the polynomial field (F_2m) are polynomials represented as bit strings, where the most-significant bit represents the highest-degree coefficient
- Significant bits of a value: Defined to be all the bits from the least significant bit up to the most significant 1 in the value. For example, the binary number 0b00010010 has 5 significant bits (0b10010).
- Significant bytes of a value: Defined to be all the bytes from the least significant byte up to the most significant non-zero byte in the value. For example, 0x00_00_02_A6_3F_00 has 4 significant bytes (0x02_A6_3F_00).
- A, B, E, N: 4096-bit values from the parameter memories (see **Section 27.6.1.10**). Unless otherwise noted, N is modulus, E is exponent or key, and A and B are other operands.
- X. Underscore denotes Montgomery format
- Data Size: Contents of the Data Size Register. Normally loaded with the number of significant *bits* in N.
- Key Size: Contents of the Key Size Register. Normally loaded with the number of significant *bytes* in E.
- Sz'(N): The value in the Data Size Register rounded up to the next multiple of 32. (This notation recognizes that Data Size is normally related to N, making Sz'(N) the number of significant bits in N, rounded up to the next multiple of 32.)
- Sz'(K): The value in the Key Size Register, times 8, rounded up to the next multiple of 32 (which is the number of bits specified by Key Size, rounded up to the next multiple of 32)

Additional notation for elliptic curve routines:

- A0, A1, A2, A3, B0, B1, B2, B3: For elliptic curve routines, 4096-bit memories A and B are each segmented into 1024-bit quantities: A = A3 • A2 • A1 • A0 and B = B3 • B2 • B1 • B0 (where • denotes concatenation).
- [X, Y] or [X, Y, Z]: coordinates of a point in two or three dimensions
- PK_BUILD: The 768-byte data structure constructed by the SPK_BUILD routine, which is the concatenation of six 1024-bit quantities: B1 • B0 • A3 • A2 • A1 • A0.

Almost all the PKEU routines require the user to supply a modulus input value. Cryptographic security depends strongly on the quality of the modulus chosen. To achieve the best security,

- In Diffie-Hellman, the modulus should be a large prime number.
- In prime field elliptic curve cryptography, the modulus should be a large prime number.
- In polynomial field elliptic curve cryptography, the modulus should be a bit string representing a large irreducible polynomial.
- in RSA, the modulus should be a product of two large prime numbers.

The Montgomery representation allows modular multiplications to take place without the costly trial division often associated with regular modular multiplication. In the Montgomery residue

number system, a number X is represented as X*R (mod N), where R = 2^s and s is computed as D*U.

The block or unit size, U, is a system parameter usually defined by the multiplier data bus size. In this version of the PKEU, its value is 32, the multiplier operand size.

The digit size D represents the quotient of the size of the modulus N (in binary representation) divided by U plus one. In simpler terms, the $2^s = 2^{DU}$ is the first multiple of 2^U larger than N.

The following sections give a brief description of the function performed by each routine using the inputs, outputs, and requirements.

27.6.1.11.1 CLEARMEMORY: Clear Memory (0x01)

- Input: Data Size (Note: this register must be loaded, but the value is not used.)
- Output: A=0, B=0, E=0, N=0
- Requirements:
  - minimum Data Size = 33
  - maximum Data Size = 4096

27.6.1.11.2 MOD_EXP: Prime field (F_p) Exponential mod N and Deconvert From Montgomery Format (0x02)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - $\underline{A}$ = field element in Montgomery format
  - E = Exponent
  - Key Size = the number of significant bytes in E
- Output: B = $\underline{A}^E$ mod N, a field element, non-Montgomery format
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - E > 1
  - maximum key (exponent) size 256 bytes
  - (number of significant bits in $\underline{A}$) < (number of significant bits in N)

Note: The input data (base) to be exponentiated must be provided in the Montgomery format. That is, $\underline{A}$ = AR (mod N), where the computation of R (the Montgomery Constant) is described in **Section 27.6.1.11**. The result is returned in parameter memory B in normal integer (non-Montgomery) representation.

27.6.1.11.3 MOD_EXP_TEQ: Exponentiate mod N and Deconvert From Montgomery Format with Timing Equalization (0x1d)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the even modulus error bit in the PKEU Interrupt Status Register is set.)
  - A = a prime field element in Montgomery format
  - E = Exponent (normal integer representation)
- Output:
  - B = A^E mod N, a prime field element, normal integer (non-Montgomery) format
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - maximum key (exponent) size 256 bytes
  - (number of significant bits in A) ≤ (number of significant bits in N)

Note: The input data (base) to be exponentiated must be provided in the Montgomery format. That is, A = AR (mod N), where the computation of R (the Montgomery Constant) is described in **Section 27.6.1.11**. The result is returned in parameter memory B in normal integer (non-Montgomery) representation.

MOD_EXP_TEQ performs the same operation as MOD_EXP but with an added timing equalization security feature. That is, its computation run-time is independent from its exponent content and is always equal to a maximum value. Also, its power consumption pattern largely follows that of an exponent with all ones.

27.6.1.11.4 MOD_R2MODN: Prime Field (F_p) Compute Montgomery Converter Constant (0x03)

- Input:
  - N = modulus. For a mathematically meaningful result, N should be odd.
  - Data Size = the number of significant bits in N
- Output: B = R^2 mod N, where R = $2^{Sz'(N)}$
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits

27.6.1.11.5 MOD_RRMODP: Prime Field (F_p) Compute Montgomery Converter Constant for Chinese Remainder Theorem (0x04)

- Input:
  - N = modulus, a prime number (modulus P or Q of CRT).
  - Data Size = the number of significant bits in N

- Key Size = number of bytes in the modulus product PQ of CRT. (Note: in this instance, Key Size is not loaded with a value related to E.)

- Output: $B = R_n R_p \bmod N$, where $R_p = 2^{Sz'(N)}$ and $R_n = 2^{Sz'(K)}$
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 1023 bits
  - minimum key size = 13 bytes
  - maximum key size = 512 bytes
  - key size > modulus size/8

27.6.1.11.6 EC_FP_AFF_PTMULT: Prime Field Elliptic Curve Scalar Point Multiply in Affine Coordinates (0x05)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - E = scalar multiplier (key)
  - Key Size = the number of significant bytes in E
  - [A0, A1] = multiplicand, an input point in affine coordinates
  - A2 = 0x1 (a bit string representing field element 1)
  - A3 = elliptic curve a parameter
  - B0 = elliptic curve b parameter
  - B1 = R^2 mod N, computed as described in MOD_R2MODN (0x03)
  - B2 = ignored
  - B3 = ignored
- Output:
  - [B1, B2] = E × [A0, A1], where × denotes elliptic curve scalar point multiplication. B1 and B2 are in affine coordinates.
  - B0 = undefined
  - B3 = undefined
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 512 bits
  - Point coordinates A0, A1 and elliptic curve parameters A3, B0 are elements of the prime field and therefore are less than the modulus N.
  - E > 1
  - maximum key size = 512 bytes

27.6.1.11.7 EC_F2M_AFF_PTMULT: Polynomial Field Elliptic Curve Scalar Point Multiply in Affine Coordinates (0x06)

- Input:

- N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
- Data Size = the number of significant bits in N
- E = scalar multiplier (key)
- Key Size = the number of significant bytes in E
- [A0, A1] = multiplicand, an input point in affine coordinates
- A2 = 0x1 (a bit string representing field element 1)
- A3 = elliptic curve a parameter
- B0 = elliptic curve c parameter:

$$c = b^{(2^{m-2})} \bmod N, \quad \text{where } m = \text{the degree of polynomial N}$$

- B1 = R^2 mod N, computed as described in F2M_R2MODN (0x0D)
- B2 = ignored
- B3 = ignored

■ Output:

- [B1, B2] = E × [A0, A1], where × denotes elliptic curve scalar point multiplication. B1 and B2 are in affine coordinates.
- B0 = undefined
- B3 = undefined

■ Requirements:

- minimum modulus size = 33 bits
- maximum modulus size = 512 bits
- Point coordinates A0, A1 and elliptic curve parameters A3, B0 are elements of the polynomial field and therefore have fewer significant bits than the modulus N.
- E > 1
- maximum key size = 512 bytes

27.6.1.11.8 EC_FP_PROJ_PTMULT: Prime Field Elliptic Curve Scalar Point Multiply in Projective Coordinates (0x07)

■ Input:

- N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
- Data Size = the number of significant bits in N
- E = scalar multiplier (key)
- Key Size = the number of significant bytes in E
- [A0, A1, A2] = multiplicand, an input point in projective coordinates
- A3 = elliptic curve a parameter
- B0 = elliptic curve b parameter
- B1 = R^2 mod N, computed as described in MOD_R2MODN (0x03)
- B2 = ignored
- B3 = ignored

- Output:
  - [B1, B2, B3] = E × [A0, A1, A2], where × denotes elliptic curve scalar point multiplication. B1, B2, and B3 are in projective coordinates.
  - B0 = undefined
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 512 bits
  - Point coordinates A0, A1, A2, and elliptic curve parameters A3, B0 are elements of the prime field and therefore are less than the modulus N.
  - E > 1
  - maximum key size = 512 bytes

27.6.1.11.9 EC_F2M_PROJ_PTMULT: Polynomial Field Elliptic Curve Scalar Point Multiply in Projective Coordinates (0x08)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - E = scalar multiplier (key)
  - Key Size = the number of significant bytes in E
  - [A0, A1, A2] = multiplicand, an input point in projective coordinates
  - A3 = elliptic curve a parameter
  - B0 = elliptic curve c parameter:

$$c = b^{(2^{m-2})} \bmod N, \quad \text{where } m = \text{the degree of polynomial } N$$

  - B1 = R^2 mod N, computed as described in F2M_R2MODN (0x0D)
  - B2 = ignored
  - B3 = ignored
- Output:
  - [B1, B2, B3] = E × [A0, A1, A2], where × denotes elliptic curve scalar point multiplication. B1, B2, and B3 are in projective coordinates.
  - B0 = undefined
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 512 bits
  - Point coordinates A0, A1, A2, and elliptic curve parameters A3, B0 are elements of the polynomial field and therefore have fewer significant bits than the modulus N.
  - E > 1
  - maximum key size = 512 bytes

27.6.1.11.10 EC_FP_ADD: Prime Field Elliptic Curve Point Add in Projective Coordinates (0x09)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - [A0, A1, A2] = first addend in projective coordinates and Montgomery format
  - A3 = elliptic curve a parameter in Montgomery format
  - B0 = elliptic curve b parameter in Montgomery format
  - [B1, B2, B3] = second addend in projective coordinates and Montgomery format
- Output:
  - [B1, B2, B3] = [A0, A1, A2] + [B1, B2, B3], where + represents an elliptic curve point addition. Outputs B1, B2, and B3 are in projective coordinates and Montgomery format.
  - B0 = elliptic curve b parameter in Montgomery format
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 512 bits
  - Point coordinates A0, A1, A2, B1, B2, B3, and elliptic curve parameters A3, B0 are elements of the prime field and therefore are less than the modulus N.

27.6.1.11.11 EC_FP_DOUBLE: Prime Field Elliptic Curve Point Double in Projective Coordinates (0x0A)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - A0 = ignored
  - A1 = ignored
  - A2 = ignored
  - A3 = elliptic curve a parameter in Montgomery format
  - B0 = elliptic curve b parameter in Montgomery format
  - [B1, B2, B3] = input point in projective coordinates and Montgomery format
- Output:
  - [B1, B2, B3] = [B1, B2, B3] + [B1, B2, B3], where + represents an elliptic curve point addition. Outputs B1, B2, and B3 are in projective coordinates and Montgomery format.
  - B0 = elliptic curve b parameter in Montgomery format
- Requirements:
  - minimum modulus size = 33 bits

- maximum modulus size = 512 bits
- Point coordinates B1, B2, B3, and elliptic curve parameters A3, B0 are elements of the prime field and therefore are less than the modulus N.

27.6.1.11.12 EC_F2M_ADD: Polynomial Field Elliptic Curve Point Add in Projective Coordinates (0x0B)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - [A0, A1, A2] = first addend in projective coordinates and Montgomery format
  - A3 = elliptic curve a parameter in Montgomery format
  - B0 = elliptic curve b parameter in Montgomery format
  - [B1, B2, B3] = second addend in projective coordinates and Montgomery format
- Output:
  - [B1, B2, B3] = [A0, A1, A2] + [B1, B2, B3], where + represents an elliptic curve point addition. Outputs B1, B2, and B3 are in projective coordinates and Montgomery format.
  - B0 = elliptic curve b parameter in Montgomery format
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 512 bits
  - Point coordinates A0, A1, A2, B1, B2, B3, and elliptic curve parameters A3, B0 are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

27.6.1.11.13 EC_F2M_DOUBLE: Polynomial Field Elliptic Curve Point Double in Projective Coordinates (0x0C)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - A0 = ignored
  - A1 = ignored
  - A2 = ignored
  - A3 = elliptic curve a parameter in Montgomery format
  - B0 = elliptic curve b parameter in Montgomery format
  - [B1, B2, B3] = input point in projective coordinates and Montgomery format
- Output:

- [B1, B2, B3] = [B1, B2, B3] + [B1, B2, B3], where + represents an elliptic curve point addition. Outputs B1, B2, and B3 are in projective coordinates in Montgomery format.
- B0 = elliptic curve b parameter in Montgomery format

- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 512 bits
  - Point coordinates B1, B2, B3, and elliptic curve parameters A3, B0 are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

27.6.1.11.14 F2M_R2: Polynomial Field (F_2m) Compute Montgomery Converter Constant (0x0D)

- Input:
  - N = modulus. For a mathematically meaningful result, N should be odd.
  - Data Size = the number of significant bits in N
- Output: $B = R^2 \bmod N$, where $R = 2^{Sz'(N)}$
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits

27.6.1.11.15 F2M_INV: Polynomial Field (F_2m) Modular Inversion (0x0E)

- Input:
  - N = modulus
  - Data Size = the number of significant bits in N
  - A = a field element
- Output: $B = A^{-1} \bmod N$, a field element
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - A is a is an element of the polynomial field and therefore has fewer significant bits than the modulus N.

27.6.1.11.16 MOD_INV: Prime Field (F_p) Modular Inversion (0x0F)

- Input:
  - N = modulus
  - Data Size = the number of significant bits in N
  - A = a field element
- Output: $B = A^{-1} \bmod N$, a field element
- Requirements:
  - minimum modulus size = 33 bits

— maximum modulus size = 4097 bits
— A is a is an element of the prime field and therefore is less than the modulus N.

27.6.1.11.17 MOD_ADD: Prime Field (F_p) Modular Addition (0x10)

- Input:
  — N = modulus
  — Data Size = the number of significant bits in N
  — A = first addend, a field element
  — B = second addend, a field element
- Output: B = (A + B) mod N, a field element
- Requirements:
  — minimum modulus size = 33 bits
  — maximum modulus size = 4096 bits
  — A and B are elements of the prime field and are therefore less than the modulus N

27.6.1.11.18 MOD_RED: Prime Field (Fp) Modulo Reduction (0x12)

- Input:
  — N = any nonzero integer
  — E = any positive integer
- Output:
  — B = E mod N, a field element modulo reduced N
- Requirements:
  — N = nonzero value
  — maximum modulus size = 256 bytes
  — Exact data_size (N) and key_size (E) must be provided

Note: E and N can be of any size, and it is not required that E > N, but N must be non-zero. This routine is mainly provided to support DSA (Digital Signature Algorithm) type algorithms where for example a 1024 bit vector must be reduced with a 160 bit modulus. This routine computes the remainder of E divided by N.

27.6.1.11.19 MOD_SUB: Prime Field (F_p) Modular Subtraction (0x20)

- Input:
  — N = modulus
  — Data Size = the number of significant bits in N
  — A = minuend, a field element
  — B = subtrahend, a field element
- Output:
  — B = (A - B) mod N, a field element
- Requirements:
  — minimum modulus size = 33 bits

- maximum modulus size = 4096 bits
- A and B are elements of the prime field and are therefore less than the modulus N

27.6.1.11.20 MOD_MULT1_MONT: Prime Field (F_p) Montgomery Modular Multiplication (0x30)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - A = multiplicand, a field element in Montgomery format
  - B = multiplier, a field element in Montgomery format
- Output:
  - B = (A x B) mod N, a field element in Montgomery format
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - Inputs A and B are elements of the prime field and are therefore less than the modulus N

27.6.1.11.21 MOD_MULT2_DECONV: Prime Field (F_p) Montgomery Modular Multiplication and Deconvert From Montgomery Format (0x40)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - A = multiplicand, a field element in Montgomery format
  - B = multiplier, a field element in Montgomery format
- Output:
  - B = (A x B) mod N, a field element, non-Montgomery format
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - Inputs A and B are elements of the prime field and are therefore less than the modulus N

27.6.1.11.22 F2M_ADD: Polynomial Field (F_2m) Modular Addition (0x50)

- Input:
  - N = modulus
  - Data Size = the number of significant bits in N
  - A = First addend, a field element
  - B = Second addend, a field element
- Output:

— B = A ⊕ B where ⊕ represents an exclusive-OR operation

- Requirements:
  - — minimum modulus size = 33 bits
  - — maximum modulus size = 4096 bits
  - — A and B are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

27.6.1.11.23 F2M_MULT1_MONT: Polynomial Field (F_2m) Montgomery Modular Multiplication (0x60)

- Input:
  - — N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - — Data Size = the number of significant bits in N
  - — A = multiplicand, a field element in Montgomery format
  - — B = multiplier, a field element in Montgomery format
- Output: B = (A x B) mod N, a field element in Montgomery format
- Requirements:
  - — minimum modulus size = 33 bits
  - — maximum modulus size = 4096 bits
  - — A and B are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

27.6.1.11.24 F2M_MULT2_DECONV: Polynomial Field (F_2m) Montgomery Modular Multiplication and Deconvert From Montgomery Format (0x70)

- Input:
  - — N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - — Data Size = the number of significant bits in N
  - — A = Multiplicand as a field element in Montgomery format
  - — B = Multiplier as a field element in Montgomery format
- Output: B = A x B mod N, a field element, non-Montgomery format
- Requirements:
  - — minimum modulus size = 33 bits
  - — maximum modulus size = 4096 bits
  - — A and B are elements of the polynomial field and therefore have fewer significant bits than the modulus N.

27.6.1.11.25 RSA_SSTEP: RSA Single Step Modular Exponentiation (0x80)

- Input:
  - — N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)

- Data Size = the number of significant bits in N
- A = a field element
- E = Exponent
- Key Size = the number of significant bytes in E

- Output: B = A^E mod N, a field element
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - E > 1
  - maximum key (exponent) size 256 bytes
  - A is an element of the field and is therefore less than the modulus N.

27.6.1.11.26 RSA_SSTEP_TEQ: RSA Single Step Modular Exponentiation with Timing Equalization (0x1e)

- Input:
  - N = modulus, an odd number (If an even number is supplied, the EVM—even modulus error bit in the PKEU Interrupt Status Register is set by the SEC.)
  - Data Size = the number of significant bits in N
  - A = a field element
  - E = Exponent
  - Key Size = the number of significant bytes in E
- Output:
  - B = A^E mod N, a field element
- Requirements:
  - minimum modulus size = 33 bits
  - maximum modulus size = 4096 bits
  - E > 1
  - maximum key (exponent) size 256 bytes

A is an element of the field and is therefore less than the modulus N.

RSA_SSTEP_TEQ performs the same operation as RSA but with an added timing equalization security feature. That is its computation run-time is independent from its exponent content and its is always equal to a maximum value. Also to a large extend its power consumption pattern follows that of an exponent with all ones.

27.6.1.11.27 SPK_BUILD: Build PK Data Structure (0xFF)

This routine must always precede any of eight elliptic curve routines, in order to prepare the data structure required by those routines.

- Input: A0, A1, A2, A3, B0, B1 = parameters required by the elliptic curve routines

- Output: PK_BUILD = B1 • B0 • A3 • A2 • A1 • A0, a structure of 6144 bits (768 bytes), (where • represents a concatenation of memory segments) In PK_BUILD, each segment is right-justified and zero-padded on the left to a total of 1024 bits
- Requirements: maximum size of each parameter = 1023 bits

27.6.2 Data Encryption Standard Execution Unit (DEU)

In typical operation, the DEU is used through channel-controlled access, which means that most reads and writes of DEU registers are directed by the SEC channels. Driver software performs core processor-controlled register accesses only on a few registers for initial configuration and error handling.

The following subsections include general descriptions of the DEU registers and structures. **Section 27.7**, *Programming Model*, on page 27-95 provides a detailed description of each register and associated register fields.

27.6.2.1 DEU Mode Register

The DEU Mode Register contains 3 bits used to program DEU operation. The Mode Register is cleared when the DEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated.

27.6.2.2 DEU Key Size Register

The key size value indicates the number of bytes of key memory to use in encrypting or decrypting. If the DEU Mode Register is set for single DES, any value other than 8 bytes automatically generates a key size error in the DEU Interrupt Status Register. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1 = K3)) or 24 bytes (168 bits for 3-key triple DES) generates an error. Triple DES always uses K1 to encrypt, K2 to decrypt, K3 to encrypt.

27.6.2.3 DEU Data Size Register

This register stores the number of bits in the final message and has an upper bound of 4096. Whatever number is written (and whatever truncated value is stored) must be a multiple of 64, except if OFB mode is selected. All data processed by the DEU must be a multiple of the DES algorithm block size of 64 bits; the DEU does not automatically pad messages out to 64-bit blocks. If a data size that is not a multiple of 64 bits is written, a data size error is generated. Only bits 7–0 are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register. This register is cleared when the DEU is reset or reinitialized.

27.6.2.4 DEU Reset Control Register

This allows three levels reset of just DEU, as defined by the three self-clearing bits:

27.6.2.5 DEU Status Register

The Status Register contains 6 fields that reflect the state of DEU internal signals. The DEU Status Register is read-only. Writing to this location results in address error being reflected in the DEU Interrupt Status Register.

27.6.2.6 DEU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the DEU Interrupt Mask Register is zero (see **Section 27.6.2.7**, *DEU Interrupt Mask Register*).

If the DEU Interrupt Status Register is non-zero, the DEU halts and the DEU error interrupt signal is asserted to the controller (see **Section 27.2.2**, *Controller Interrupts*). In addition, if the DEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the DEU Reset Control Register[RI] (see **Section 27.7.9.4**, *AESU Reset Control Register (AESURCR)*, on page 27-235).

27.6.2.7 DEU Interrupt Mask Register

The Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.6.2.6**, *DEU Interrupt Status Register*, on page 27-43), if the corresponding bit in this register is set, then the error is ignored; no bit is set in the DEU Interrupt Status Register, and no error interrupt occurs. If the corresponding bit is not set, then upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.2.8 DEU End_of_Message Register

Writing the end-of-message register is a handshake mechanism indicating the end of incoming data. DESA signals a done interrupt when the input FIFO becomes empty after the End-of-Message is issued. After the final message block is written to the input FIFO, the End_of_Message register must be written. The value in the Data Size Register is used to determine how many bits of the final message block (always 64) to process. Note that the End_of_Message register has no data size, and during the write operation, the core processor data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful: it always returns a zero value

and does not generate any error. Writing to this register is the trigger that causes the DEU to process the final block of a message, allowing it to issue a done interrupt.

27.6.2.9 DEU IV Register

For CBC mode, the initialization vector is written to and read from the DEU IV register. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading this memory location while the module is processing data generates an error interrupt.

27.6.2.10 DEU Key Registers

The DEU uses three write-only key registers, K1, K2, and K3, to perform encryption and decryption. In Single DES mode, only K1 is written and the value is simultaneously written to K3, auto-enabling the DEU for 112-bit Triple DES if the Key Size Register indicates 2-key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit Triple DES, K1 must be written first, followed by the write of K2, then K3. Reading any of these memory locations generates an address error interrupt.

27.6.2.11 DEU FIFOs

The DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the DEU FIFO address space enqueues data to the DEU input FIFO, and a read from anywhere in the DEU FIFO address space dequeues data from the DEU output FIFO.

Writes to the input FIFO go first to a staging register that can be written in byte, 4-byte, or 8-byte units. When all 8 bytes of the staging register are written, the entire 8 bytes is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. Since the DEU data length should always be a multiple of 8 bytes, the last write should complete the 8-byte set. However, if there is any partial data set in the staging register when the DEU End_of_Message Register is written, the partial data set is automatically padded with zeros to a full 8 bytes and enqueued to the input FIFO.

The output FIFO is readable in byte, 4-byte, or 8-byte units. When all 8 bytes of the header are read, that 8-byte set is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflow caused by reading or writing the DEU FIFOs are reflected in the DEU Interrupt Status Register.

27.6.3 Advanced Encryption Standard Execution Unit (AESU)

In typical operation, the AESU is used through channel-controlled access, which means that most reads and writes of AESU registers are directed by the SEC channels. Driver software performs core processor-controlled register accesses only on a few registers for initial configuration and error handling.

This EU includes an ICV checking feature, that is, it can generate an integrity check value (ICV) and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the core processor either via interrupt by a writeback of EU status fields into core processor memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-196), and mask the ICE bit in the Interrupt Mask Register (**Section 27.6.3.7**, *AESU Interrupt Mask Register*, on page 27-47). In this case the normal done signaling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, there is an error interrupt generated to the core processor, but no done interrupt or writeback.

The following subsections include general descriptions of the AESU registers and structures. **Section 27.7**, *Programming Model*, on page 27-95 provides a detailed description of each register and associated register fields.

27.6.3.1 AESU Mode Register

The AESU Mode Register contains 7 fields used to program the AESU. The Mode Register is cleared when the AESU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated.

In most networking applications, the decryption of an AES protected packet is performed as a single operation. However, if circumstances require that the decryption of a message be split across multiple descriptors, the AESU allows the user to save the decrypt key and the active AES context to memory for later reuse. This eliminates the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data decrypted).

27.6.3.2 AESU Key Size Register

The AESU Key Size Register stores the number of bytes in the key (16, 24, or 32). Any key data beyond the number of bytes in the Key Size Register is ignored. This register is cleared when the

AESU is reset or reinitialized. If you specify a key size other than 16, 24, or 32 bytes, an illegal key size error is generated. If the Key Size Register is modified during processing, a context error is generated.

27.6.3.3 AESU Data Size Register

The AESU Data Size Register stores the number of bits in the final message block. Acceptable sizes vary depending on the AES mode selected. In ECB and CBC modes, the message processed by the AESU must be a multiple of 128 bits; the AESU does not automatically pad messages out to 128-bit blocks. In CCM and CTR modes, data size must be a multiple of 8 bits. In XOR mode the data size must be a multiple of 256 bits (32 bytes). If an improper data size is written, a data size error is generated. Only the lowest 3, 7, or 8 bits of the Data Size Register are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register. This register is cleared when the AESU is reset or reinitialized. Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

27.6.3.4 AESU Reset Control Register

This register allows three levels reset of just AESU, as defined by the three self-clearing bits:

27.6.3.5 AESU Status Register

AESU Status Register is a read-only register that reflects the state of six status outputs. Writing to this location result in an address error being reflected in the AESU Interrupt Status Register.

27.6.3.6 AESU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the AESU Interrupt Mask Register is zero (see **Section 27.6.3.7**, *AESU Interrupt Mask Register*, on page 27-47).

If the AESU Interrupt Status Register is non-zero, the AESU halts and the AESU error interrupt signal is asserted to the controller (see **Section 27.2.2**, *Controller Interrupts*). In addition, if the AESU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the AESU Reset Control Register [RI] bit.

27.6.3.7 AESU Interrupt Mask Register

The AESU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.7.9.6**, *AESU Interrupt Status Register (AESUISR)*, on page 27-237), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.3.8 AESU End_of_Message Register

The AESU End_of_Message register indicates whether an AES operation is completed. After the final message block is written to the input FIFO, you must write to the End_of_Message register. The value in the Data Size Register is used to determine how many bits of the final message block (always 128) to process. Writing to this register causes the AESU to process the final block of a message, allowing it to signal a done interrupt. A read of this register always return a zero value.

27.6.3.9 AESU Context Registers

There are twelve 64-bit context data registers that allow the core processor to read/write the contents of the context used to process the message. The context must be written prior to the key data. If the Context Registers are written during message processing, a context error is generated. All Context Registers are cleared when a hard/soft reset or initialization is performed. If a message is processed through the AESU in two separate operations (for example, using two descriptors), then the context must be read out of the SEC at the end of the first operation and then restored at the beginning of the second operation.

Not all of the AESU context registers are used in every mode. However, context is always read and restored as a contiguous subset of the twelve context registers ending with the highest numbered register used in that cipher mode. The context registers are summarized by modes in **Table 27-2**, **Table 27-3**, and **Table 27-4**.

Table 27-2 summarizes the context registers used for confidentiality modes only.

Table 27-2. AESU Context Registers for Confidentiality Modes

Context Register (address offset)	Cipher Mode Providing Only Confidentiality			
	CBC/ CBC-RBP/ OFB/ CFB128 (Section 27.6.3.9.1)	CTR (Section 27.6.3.9.2)	SRT (Section 27.6.3.9.3	XTS (Section 27.6.3.9.4)
1 (0xC4100)	IV *		Counter *	I *
2 (0xC4108)				sector size *
3 (0xC4110)	—	—	Counter Modulus *	—
4 (0xC4118)			—	
5 (0xC4120)	—	Counter *	—	—
6 (0xC4128)				

Table 27-2. AESU Context Registers for Confidentiality Modes

Context Register (address offset)	Cipher Mode Providing Only Confidentiality			
	CBC/ CBC-RBP/ OFB/ CFB128 (Section 27.6.3.9.1)	CTR (Section 27.6.3.9.2)	SRT (Section 27.6.3.9.3	XTS (Section 27.6.3.9.4)
7 (0xC4130)	—	Counter Modulus Exponent *	—	—

Notes: **1.** Context Registers 8 through 12 are unused for these modes
2. * = Must be written at start of new message, except if zero.
3. —- = ignored.

Table 27-3 summarizes the different cipher modes that provide data integrity only.

Table 27-3. AESU Context Registers for Integrity Modes

Context Register (address offset)	Cipher Mode Providing Only Data Integrity		
	XCBC-MAC (Section 27.6.3.9.5)	GCM-GHASH (Section 27.6.3.9.6)	CMAC (Section 27.6.3.9.7)
1 (0xC4100)	Computed MAC	Computed MAC	Computed MAC
2 (0xC4108)			
3 (0xC4110)	Received MAC*	—	Received MAC*
4 (0xC4118)			
5 (0xC4120)	$E(K, 0^{128})$	—	Key 3
6 (0xC4128)			
7 (0xC4130)	—	$len(AAD)^T$	Key 2
8 (0xC4138)		—	
9 (0xC4140)	—	H	Key 1**
10 (0xC4148)			
11 (0xC4150)	—	$len(AAD)^C$	—

Notes: **1.** Context register 12 is unused for these modes.
2. * Used only in ICV mode. Must be written at start of new message for ICV checking.
3. ^{T} = length of total data (in bits)
4. ** = Output only, not reloaded
5. ^{C} = length of data processed with current descriptor (in bits)
6. — = ignored

Table 27-4 summarizes the context registers used for confidentiality and integrity.

Table 27-4. AESU Context Registers for Modes Providing Confidentiality and Integrity

Context Register (address offset)	Cipher Mode Providing Confidentiality and Integrity	
	CCM (Section 27.6.3.9.8)	GCM (Section 27.6.3.9.9)
1 (0xC4100)	IV* / MAC	Computed MAC
2 (0xC4108)		
3 (0xC4110)	Encrypted MAC** / Decrypted MAC / Encrypted Counter	Received MAC
4 (0xC4118)		
5 (0xC4120)	Counter*	Counter
6 (0xC4128)		
7 (0xC4130)	Counter Modulus Exponent* (header size/ MAC size)***	$len(AAD)^T$
8 (0xC4138)	—	$len(IV)^T$
9 (0xC4140)	—	Y_0
10 (0xC4148)		
11 (0xC4150)	—	$len(AAD)^C$
12 (0xC4158)		$len(IV)^C$

Notes:
1. * = Must be written at start of new message, except if zero
2. ** = Must be written at start of new CCM decryption
3. *** = The Header and MAC Sizes are internally constructed by the AES engine; then, that information is included inside Context Register 7 for context switching purposes
4. ^{C} = length of data processed with current descriptor (in bits)
5. ^{T} = length of total data (in bits)
6. ^{ICV} = Needed only in ICV mode
7. — = Ignored

27.6.3.9.1 Context for CBC, CBC-RBP, OFB, and CFB128 Cipher Modes

Within the Context registers, for use in CBC, CBC-RBP, OFB, and CFB128 cipher modes, are two 64-bit context data registers that allow the core to read/write the contents of the initialization vector (IV):

- Context Register 1 holds the ***least*** significant bytes of the initialization vector (bytes 1–8).
- Context Register 2 holds the ***most*** significant bytes of the initialization vector (bytes 9–16).

The IV must be written prior to the message data. If the IV registers are written during message processing, or the mode is not set, a context error will be generated.

The IV registers may only be read after processing has completed, as indicated by the assertion of DI (done interrupt) in the AESU status register as shown in Section 27.7.9.5, **AESU Status Register (AESUSR)**". If the IV registers are read prior to assertion of Interrupt Done, an early read error will be generated.

27.6.3.9.2 Context for Counter (CTR) Cipher Mode

In counter cipher mode, a random 128-bit initial counter value is incremented modulo 2^M with each block processed. The running counter is encrypted and XORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent M can be set between 8 and 128 in multiples of 8.

There are two options for loading CTR mode context. When using descriptor type 0001_0, AES-CTR context is loaded as shown in **Table 27-2**. The Context_IN length must be set to 56 B, and the context itself must be 32 B of zeroes (context registers 1–4), followed by the initial counter value (context registers 5–6), and finally, the modulus exponent M in context register 7. When using descriptor type 0000_0, 24 B of context can be loaded as shown for SRT, dispensing with the need for the initial zeroes.

27.6.3.9.3 Context for SRT Cipher Mode

As was noted for the AESU Mode Register, SRT is not a new AES cipher mode, it is an AESU method of performing AES-CTR cipher mode with reduced context loading overhead. This mode as originally developed for SRTP, but is also applicable to the use of AES-CTR for LTE EEA2. SRT cipher mode can be used with a descriptor type 0010_0 (SRTP). As with CTR cipher mode, a random 128-bit initial counter value is incremented modulo 2^M with each block processed. The running counter is encrypted and XORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent M can be set between 8 and 128 in multiples of 8.

As shown in **Table 27-2**, in SRT mode, context registers 1–2 hold the initial counter value, and context register 3 holds the modulus exponent M.

Note: There are two methods of performing AES-CTR with reduced context loading. Either use descriptor type 0000_0 with the AES mode register set to CTR, or use descriptor type 0001_0 with the AES mode register set to SRT. These methods are completely equivalent for cipher only operations (no snooping for ICV generation). When performing AES-CTR in conjunction with HMAC-SHA-1, reduced context loading can only be achieved by using SRT.

27.6.3.9.4 Context and Operation for XTS Cipher Mode

IEEE P1619 describes XTS mode as a tweakable block cipher used for encryption of sector-based storage. The key material for XTS consists of a data encryption key (used by the AES block cipher) as well as a tweak key that is used to incorporate the logical position of the data block into the encryption. The key is parsed as a concatenation of two fields of equal size called Key1 and Key2 (16 or 32 bytes each).

A 256-bit or 512-bit key is associated with an ordered sequence of sectors, numbered consecutively. The sequence of sectors that are associated with the key is related to the scope of

that key. In order to encrypt or decrypt a sector, the sequence number of this sector within the scope of the key (I) must be known. Each 16-byte block within the sector has its own sequence number that gives the logical position of the data block inside the sector.

The tweak value (T) is computed based on both the sector number (I) and the 16-byte block number within the sector (j) as

T_0 = aes_encrypt(I, Key2);

T_j = $xtime^j(T_0)$;

The *xtime* function can be described mathematically as follows:

If L(127) = 0, the xtime(l) = L << 1; Else xtime(L) = (L << 1) XOR 0x87;

where L is a 128-bit vector and L[127] is its most significant bit. The irreducible polynomial $f = \alpha^{128} + \alpha^7 + \alpha^2 + \alpha + 1$ is used in the tweak calculation and can be represented as 0x87.

When the last block of the message is not a full 16-byte block, processing of the penultimate and the last block implements a borrowing mechanism whereby bits from the penultimate block ciphertext are appended to the last block plaintext to pad it out to a 16-byte boundary. This is described in detail in **IEEE** Std. P1619™.

For sector byte sizes that are divisible by 16, XTS mode also supports processing of multiple sectors per session where the sector sequence number (I) is automatically incremented. When multiple sectors are decrypted, the tweak key (Key2) is expanded once for the initial tweak computation pertaining to the first sector; this expanded value is directly used for other sectors. In case that a message needs to be processed in multiple XTS sessions, message splitting must be on a sector boundary.

XTS cipher mode uses context register 1 for the index (I) and context register 2 for the sector size (see **Table 27-2**).

For core-controlled operation of the AESU in XTS cipher mode, the following steps must be performed:

1. Reset
2. Write the Mode Register to:
   a. Set Cipher Mode to XTS
   b. Specify encryption/decryption
3. Load Key
4. Load I into context register 1 and the sector size in bytes into context register 2. Note that I must be written as big-endian (LSB in the left-most bit positions), while sector size must be in little-endian format.

5. Set Key size
6. Set Data size
7. While available:
   a. Load plaintext (for encryption) or ciphertext (for decryption) blocks
   b. Unload ciphertext (for encryption) or plaintext (for decryption) blocks
8. Write to the End of Message register
9. Unload final ciphertext (for encryption) or plaintext (for decryption) blocks

27.6.3.9.5 Context and Operation for XCBC-MAC Cipher Mode

XCBC-MAC cipher mode is an authentication only mode of AES. Normal CBC-MAC runs AES in CBC cipher mode and assigns the final ciphertext result as the MAC. XCBC-MAC supports only 16-byte keys.

The AESU supports three mode options while operating in XCBC-MAC cipher mode. These options are controlled by the AUX bits in the AESU Mode Register. The encrypt/decrypt bit has no meaning in an XCBC-MAC operation and is ignored by the AESU. The AUX bits are as follows:

- *AUX0*. Controls whether the XCBC-MAC is completed with this descriptor, or whether this descriptor is only doing a portion of the total MAC generation, and context needs to be output so that it can be reloaded for subsequent operations.
- *AUX1*. Controls whether K1, K2, and K3 are initialized at the start of the descriptor, or whether previously initialized keys are reloaded.
- *AUX2*. Controls whether the AESU performs automatic checking of a received MAC against the newly calculated MAC.

For a descriptor that generates an XCBC-MAC over a full message, program AUX0 = 0, AUX1 = 0, and AUX2 = 0. The descriptor key length is loaded into the AESU Key Size Register, and the key itself is loaded into the AESU Key Registers IU and IL. The generated MAC is held in the AESU Context Registers 1 and 2 and output according to the ICV Output length and pointer in the descriptor.

For a descriptor that generates an XCBC-MAC over a full message and compares the calculated MAC with the MAC received with the message, program AUX0 = 0, AUX1 = 0, and AUX2 = 1. The descriptor key length is loaded into the AESU Key Size Register, and the key itself is loaded into the AESU Key Registers IU and IL. The generated MAC is held in AESU Context Registers 1 and 2 and is compared to the received MAC that the channel loads into AESU Context Registers 3 and 4 using the Extent field in the descriptor. Success or failure of the MAC comparison can be reported by an interrupt or through the ICCR1 bits in the modified descriptor header, if header writeback is enabled.

Sometimes a message cannot be processed by a single descriptor. All the data might not be present, or the message might be larger than a single XCBC-MAC descriptor can process (>64KB-1). In either case, this situation can be handled through combinations of AUX mode bits.

For the first descriptor that processes the initial portion of the message, program AUX0 = 1, AUX1 = 0, and AUX2 = 0. The descriptor key length is loaded into the AESU Key Size Register and the key itself is loaded into the AESU Key Registers 1U and 1L. The AESU generates K3, K2, and K1 and stores them, respectively, in AESU Context Registers 5-6, 7-8, and 9-10. When the first descriptor has consumed all of its message data, it outputs the contents of Context Registers 1–10 using the Context Out length (80B) and pointer.

There can be an unlimited number of intermediate descriptors that are neither the first nor last descriptor. Intermediate descriptors program AUX0 = 1, AUX1 = 1, and AUX2 = 0. The descriptor key length is set to 16B and the key pointer is set to the address of Key 1 (Context Registers 9–10) from the previous descriptor. This loads Key 1 into AESU Key Size Registers 1U and 1L. The descriptor Context In length is set to 64B, and the pointer is set to the address of Context Registers 1–8 from the previous descriptor.

When an intermediate descriptor has consumed all of its message data, it outputs the contents of Context Regs 1–10 using the Context Out length (80B) and pointer.

For the last descriptor that processes the final portion of the message, program AUX0 = 0, AUX1 = 1, and AUX2 = 0. The descriptor's key length is set to 16B and the key pointer is set to the address of Key 1 (Context Registers 9–10) from the previous descriptor. This loads Key 1 into AESU Key Size Registers 1U and 1L. The descriptor Context In length is set to 64B, and the pointer is set to the address of Context Registers 1–8 from the previous descriptor.

When the last descriptor has consumed all its message data, the generated MAC is held in AESU Context Regs 1–2 and output according to the ICV Out length and pointer in the descriptor. To compare the calculated MAC with the MAC received with the message, program AUX2 = 1 on the final descriptor. The generated MAC is held in AESU Context Regs 1–2, and is compared to the received MAC, which is input using the Extent field in the descriptor. The channel overwrites the context reload values in Context Regs 3–4 with the real received MAC.

27.6.3.9.6 Context and Operation for GCM-GHAS Cipher Mode

GCM-GHAH denotes the authentication part of GCM cipher mode and is described in **Section 27.6.3.9.9**, *Context and Operation for GCM Cipher Mode*, on page 27-58.

27.6.3.9.7 Context and Operation for CMAC (OMAC1) Cipher Mode

CMAC cipher mode is an authentication only mode of AES. CMAC can be specified using the following notation:

- E(K,L) denotes the AES-encrypt function;
- xtime(L) is defined as follows, where L is a 128-bit vector with L[127] as most significant bit:
  - If L[127]=0, then xtime(L)=L<<1 (where '<<' denotes bitwise left shift)
  - Else xtime(L) = (L<<1) XOR 0x87.

The AESU supports three mode options while operating in CMAC mode. These options are controlled by the AUX bits in the AESU Mode Register. The encrypt/decrypt bit has no meaning in a CMAC operation and is ignored by the AESU.

- *AUX0*. Controls whether the CMAC is completed with this descriptor or whether this descriptor is only doing a portion of the total MAC generation, and context needs to be output so that it can be reloaded for subsequent operations.
- *AUX1*. Controls whether K1 and K2 are initialized at the start of the descriptor or whether previously initialized keys are reloaded.
- *AUX2*. Controls whether the AESU must perform automatic checking of a received MAC against the newly calculated MAC.

For a descriptor that generates a CMAC over a full message, program AUX0 = 0, AUX1 = 0, and AUX2 = 0. The descriptor key length is loaded into the AESU Key Size Register, and the key itself is loaded into the AESU Key Registers. The generated MAC is held in AESU Context Registers 1–2, and output according to the ICV Out length and pointer in the descriptor.

Note: For some uses of CMAC, the message data itself can be modified to include packet specific context in the CMAC generation process. For instance, when using AES-CMAC for the LTE EIA2 algorithm, a 64b IV-like value is prepended to the PDCP header prior to calculating the MAC. The 64b value consists of COUNT (32b)||Bearer (5b)|| Direction (1b) || Zeroes (26b).

For a descriptor that generates a CMAC over a full message and compares the calculated MAC with the MAC received with the message, program AUX0 = 0, AUX1 = 0, and AUX2 = 1. The descriptor key length is loaded into the AESU Key Size Register, and the key itself is loaded into the AESU Key Registers. The generated CMAC is held in AESU Context Registers 1–2, and is compared to the received CMAC, which the channel loads into AESU Context Registers 3–4 using the Extent field in the descriptor. Success or failure of the MAC comparison can be reported by an interrupt or through the ICCR1 bits in the modified descriptor header if header writeback is enabled.

Sometimes a message cannot be processed by a single descriptor. All the data might not be present, or the message might be larger than a single CMAC descriptor can process (>64KB – 1). In either case, this situation can be handled through combinations of AUX mode bits.

For the first descriptor that processes the initial portion of the message, program AUX0 = 1, AUX1 = 0, and AUX2 = 0. The descriptor key length is loaded into the AESU Key Size Register, and the key itself is loaded into the AESU Key Registers. The AESU internally generates CMAC context and stores it respectively in AESU Context Registers 5–6. When the first descriptor has consumed all of its message data, it outputs the contents of Context Registers 1–6 using the Context Out length (48B) and pointer.

There can be an unlimited number of intermediate descriptors that are neither the first nor last descriptor. Intermediate descriptors program AUX0 = 1, AUX1 = 1, and AUX2 = 0. The descriptor key length is set to 0. The descriptor Context In length is set to 48B and the pointer is set to the address of Context Registers 1–6 from the previous descriptor.

When an intermediate descriptor has consumed all of its message data, it outputs the contents of Context Registers 1–6 using the Context Out length (48B) and pointer.

For the last descriptor that processes the final portion of the message, program AUX0 = 0, AUX1 = 1, and AUX2 = 0. The descriptor key length is set to 0. The descriptor Context In length is set to 48B, and the pointer is set to the address of Context Registers 1–6 from the previous descriptor.

When the last descriptor has consumed all of its message data, the generated CMAC is held in AESU Context Registers 1–2 and output according to the ICV Out length and pointer in the descriptor. To compare the calculated MAC with the MAC received with the message, program AUX2 = 1 on the final descriptor. The generated CMAC is held in AESU Context Regs 1–2, and is compared to the received CMAC, which is input using the Extent field in the descriptor. The channel overwrites the context reload values in Context Regs 3–4 with the real received MAC.

27.6.3.9.8 Context for CCM Cipher Mode

The SEC AESU can performing single pass encryption and MAC generation. The core processor is required to arrange the CCM context so that the context can be fetched as a contiguous string into the Context Registers prior to encryption/MAC generation or decryption/MAC validation. The Context Register contents for CCM mode is summarized in **Figure 27-6**.

		Context Registers						
		1	2	3	4	5	6	7
Encrypt (outbound)	Inputs	IV		0		Initial Counter		Counter Modulus Exponent
	Outputs	MAC	0	MIC	0			
Decrypt (inbound)	Inputs	IV		MIC	0	Initial Counter		Counter Modulus Exponent
	Outputs	Computed MAC	0	Decrypted MAC	0			

Figure 27-6. AESU CCM Context Registers

Note: AES-CCM mode does not support zero length AAD and zero length payload simultaneously. Either AAD length or payload length must be greater than zero.

The context for CCM encryption/MAC generation is:

- Reg 1–2 are session specific and hold the 128-bit Initialization Vector (from memory)
- Reg 3–4 contains 128 bits of zero padding
- Reg 5–6 are a session specific counter (Initial Counter Value) (from memory)
- Reg 7 has the Counter Modulus Exponent (msb to lsb). Should be fixed at 0x0000_0080.

Note: The counter modulus for CCM mode is currently defined as 2^{128} making the exponent 128. This value is made programmable in the SEC in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU Context Register prior to CCM encryption.

- *CCM Encryption Processing*. With the session specific key and context, the AESU performs the following operations.

1. Initialize the IV, and encrypt with the symmetric key.
2. In CBC fashion, take the output of step 1, hash with the first block of plaintext, and encrypt with the symmetric key.
3. Continue as in step 2 until the final block of plaintext is processed. The result of the encryption of the final block of plaintext with the symmetric key is the MAC Tag. The

full 128bits of MAC data is written to Context Registers 1-2, for use in the next phase of CCM processing.

4. Once the MAC Tag is generated (step 3), the MAC tag, along with the plaintext is encrypted with the AESU operating in Counter mode.
5. The first item to be encrypted in Counter Mode is the counter (Initial Counter Value) from Context Registers 5–6. The counter is encrypted with the symmetric key, and the result is hashed with the MAC Tag (retrieved from Context Registers 1–2) to produce the MIC (encrypted MAC), which is then stored in Context Registers 3-4. At the completion of CCM encrypt processing, this MIC is output to memory (per the descriptor pointer) for the core processor to append to the **IEEE** 802.11i standard frame. Note: The MIC written out to memory by the AESU is the full 128 bits. The core processor must only append the most significant 64 bits to the frame as the MIC.
6. The counter value is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of plaintext to produce the first block of cipher text. The ciphertext is placed in the AESU output FIFO.
7. The counter continues to be incremented, and encrypted with the symmetric key, with the result hashed with each successive block of plaintext, until all plaintext is converted to ciphertext. The SEC controller manages FIFO reads and writes, fetching plaintext and writing ciphertext per the pointers provided in the descriptor. When all ciphertext and the MIC is output, the CCM encrypt operation is complete.

- *CCM Decryption Processing*. The context for CCM decryption/MAC generation is:
  - Reg 1–2 is session specific and holds the 128 bit Initialization Vector (from memory)
  - Reg 3–4 holds the MIC (from the received frame) + 64 bits of zero padding
  - Reg 5–6 is a session specific counter (Initial Counter Value) (from memory)
  - Reg 7 holds the Counter Modulus Exponent (msb to lsb). Should be fixed at 0x0000_0080.

Note: The counter modulus for CCM mode is currently defined as 2^128, making the exponent 128. This value is made programmable in the SEC to in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU Context Register prior to CCM decryption.

CCM decryption processing is the reverse of encryption. With the session specific key and context, the AESU performs the following operations.

1. Initializes the IV, and encrypts with the symmetric key. Simultaneously, the counter (Initial Counter Value) from Context Registers 5–6 is encrypted with the symmetric key. The result is hashed with the Encrypted MAC (from Context Register 3–4), and the

resulting Original MAC is written to Context Registers 3–4, overwriting the encrypted MAC.

Note: Strictly speaking, the Counter is encrypted with the symmetric key, however the AESU should be set for decrypt to perform the counter and CBC processes in the correct order.

2. The **IEEE** Std. 802.11 frame header is hashed with the encrypted IV. (The AESU automatically determines the header length.) Simultaneously, the counter is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of ciphertext to produce the first block of plaintext. The plaintext is placed in the AESU output FIFO, while simultaneously, in CBC fashion, a copy of the first block of plaintext is hashed with the output of encryption of the **IEEE** Std. 802.11 frame header. The output is encrypted with the symmetric key.
3. As each ciphertext block is converted to plaintext, the plaintext is CBC encrypted. When the final plaintext block is processed, the CBC MAC (MAC Tag) is written to Context Registers 1–2. The first 64 bits of the MAC Tag are compared to the MAC Tag recovered in step 1.

Note: For both encrypt and decrypt operations, if the **IEEE** 802.11 frame is being processed as a whole (not split across multiple descriptors), the Initialize and Final MAC bits should be set in the AESU Mode Register.

27.6.3.9.9 Context and Operation for GCM Cipher Mode

Galois Counter Mode (GCM) uses AES Counter mode to achieve data confidentiality. Authentication is achieved by computing a GHASH Message Authentication Code (GMAC) through performing repetitive multiplication-accumulate functions in a Galois Field.

Normally, the IV (which is provided through the input FIFO) is 96 bits. If it is 96 bits, then the IV is padded with the value $(\{0\}^{31}1)$, where $(0)^{31}1$ is defined as a string of 31 bits of 0 followed by a single bit of 1. Otherwise, the IV is hashed using the GHASH (H, {}, IV) function, where H is defined as $E(\{0\}^{128}, K)$, E stands for encryption operation, and K represents the key used. The resulting value Y_0 (the padded IV or the GHASHed IV) is provided as the counter input to Counter Mode AES. The result of encrypting Y_0 is called $E(Y_0, K)$, and is used to generate the final MAC tag.

Data is encrypted or decrypted by XORing input data with the Pseudorandom Key Stream generated by Counter Mode AES, starting with the second PRK block. Initial counter value Y_0 is incremented modulo 2^{32}.

GCM cipher mode can be used to perform only the authentication part (GHASH (H, AAD, ciphertext)). To do this, set AUX0 and specify encryption operation. This special sub-mode is called GCM-GHASH in this document. The format of the context registers for GCM-GHASH mode is shown in **Table 27-10** on page 27-65. GCM cipher mode also has option of

automatically verifying that the received and computed MAC tags are identical. This cipher mode is called GCM w/ICV and can be specified by setting Mode Register bits 56, 57 and 62 to 1, and bit 61 to 0. GCM w/ICV context format is shown in **Table 27-9** on page 27-64.

Messages (IV+AAD+textdata) are fed in through the input FIFO, and are always processed in the following order: IV, AAD, textdata, followed by the final MAC computation (where “textdata” refers to plaintext or ciphertext to be operated on). The whole message, however, does not have to be processed in one GCM execution. It can be split and processed in multiple GCM runs separated by resets of the AESU block i.e. in multiple descriptors. The boundaries can be set at the end of any full block (16 bytes) of the stream IV+AAD+textdata. Hence, any of the individual components (IV, AAD, or textdata) can be split into multiple descriptors. Refer to 27-5 for proper AUX mode specification in this case and to **Table 27-7** through **Table 27-10** for proper context formatting. It should be noted that in case of a late arrival of the MAC tag on the receiving side, the final MAC can be computed and verified against the received MAC in a separate descriptor after the rest of the message (IV+AAD+textdata) has already been processed.

For core-controlled operation of the AESU in GCM cipher mode, perform the following steps:

1. Reset.
2. Set Cipher Mode to GCM or GCM w/ICV and specify encrypt, decrypt in the Mode Register. To perform GCM-GHASH (only GHASH (H, AAD, ciphertext) is computed) set AUX0 and specify encrypt. Set AUX2 and AUX1 bits according to 27-5.
3. Load Key.
4. Load (Restore) Context as needed (see 27-7 to 27-10).
5. Set Key size.
6. Set the size of the computed/received MAC (8, 12 or 16 bytes, default is 16).
7. Set Data size.
8. While available:
   a. Load IV into the input FIFO (1 or multiple blocks up to 2^{64} bits in total).
   b. Load AAD into the input FIFO (0 or multiple blocks up to 2^{64} bits in total).
   c. Load plaintext (for encryption) or ciphertext (for decryption) blocks into the input FIFO.
   d. Unload ciphertext (for encryption) or plaintext (for decryption) blocks from the output FIFO.
9. Write to the End of Message register.
10. Unload final ciphertext (for encryption) or plaintext (for decryption) blocks.
11. Read (Save) context registers if another segment of the message will be processed later.
12. Read final GCM MAC from Context Registers 1–2, if AUX2 bit was set in Mode Register.
13. For GCM w/ICV, check ICCR bits in the AESU Status Register.

Table 27-5. GCM Cipher Mode Auxiliary Bit Definitions

Auxiliary Bit	Definitions	
	0	1
AUX2 (bit 58)	Do not compute MAC	Compute MAC
AUX1 (bit 59)	One of the following cases: • Descriptor contains the whole message (IV+AAD+textdata) • Descriptor contains the whole IV and no or part of AAD or textdata • Descriptor contains a non-final part of IV, AAD, textdata (IV, AAD or textdata split between descriptors) • Descriptor contains the final part of AAD or textdata but no MAC is computed	One of the following cases: • Descriptor contains the final part of IV (IV split between descriptors) - $len(IV)^T$ needed • Descriptor contains the final part of textdata and the final MAC is computed, i.e. AUX2=1 (textdata split between descriptors) - $len(AAD)^T$, $len(textdata)^T$ needed • Descriptor contains the whole textdata but no or part of AAD and the final MAC is computed - $len(AAD)^T$, $len(textdata)^T$ needed • Descriptor contains the final part of AAD and the final MAC is computed - $len(AAD)^T$, $len(textdata)^T$ needed • Descriptor computes only MAC (based on restored context) but does not contain either IV, AAD or textdata - $len(AAD)^T$, $len(textdata)^T$ needed
AUX0 (bit 60) and Encrypt	—	GHASH-only mode
AUX0 (bit 60) and Decrypt	The key is to be unrolled	The key is already unrolled

AUX0 has different use depending on whether encryption or decryption is specified. For decryption, it determines whether the provided key should be first unrolled before processing starts, while in case of encryption it should generally be set to 0 unless GCM-GHASH cipher mode is desired. AUX2 bit determines whether the final MAC tag is to be computed or not. If AUX2 is set to 1, $E(K, Y_0)$ and the last iteration of the GHASH(H, AAD, ciphertext) is going to be performed and then XORed to give the MAC tag. Hence, if the message is split into multiple descriptors, only the last one should have AUX2=1 for proper MAC tag computation. AUX1 is used to resolve the issues related to the splitting of messages into multiple descriptors. **Table 27-5** shows the proper settings of AUX1 for several scenarios of message splitting. In general, whenever the final GHASH iteration needs to be computed (either for GHASH(H, {}, IV) or GHASH(H, AAD, ciphertext)), and the current length is not equal to total length for either IV, AAD, or textdata, AUX1 should be set to 1. Consequently, an AUX1 value of 1 also indicates that the context registers 9–10 need to provide the total length of IV, AAD, or textdata for this to be accomplished. **Table 27-6** describes how the context registers are used for GCM processing.

Table 27-6. Description of Context Register for GCM Processing

Registers	Description
1–2	Intermediate MAC value. This needs to be provided only when switching context during AAD and/or textdata processing (AAD+textdata stream split into multiple descriptors). On the output side, these registers contain either intermediate MAC tag in case of context switching (requires AUX2=0) or the final MAC tag at the end of processing if AUX2=1. If AUX2=0 on the last descriptor processing a particular message, these registers will contain partially computed GHASH(H, AAD, ciphertext) where the last GHASH iteration is not computed. In case of GCM w/ICV, the final MAC tag written here as the result of GCM processing will be truncated to 8, 12, or 16 (no truncation) bytes as defined in ICV Size register. Note, that any size from 1 to 16 bytes can be specified in ICV Size register but any value other than 8 or 12 will be defaulted to 16 bytes automatically.
3–4	Received MAC tag used only in case of inbound processing with GCM w/ICV. This can be a 8, 12 or 16-byte block as specified by writing to the ICV Size register.
5–6	Counter value Y_i is required only if restoring context to continue processing a message. Note that the same value read when saving context should be written to these registers when restoring context since it is automatically incremented after every processed block. In case of GCM-GHASH, these registers are not used.
7	Total length of AAD in bits. This is the total AAD length irrespective of whether AAD is split in multiple descriptors. It is required when AUX1=1 and the current descriptor processes the last segment of AAD or textdata. It is also required if the whole message is already processed and the current descriptor only computes the final MAC tag.
8	Total length of the plaintext/ciphertext or IV in bits. Required only when AUX1=1 (see 27-5). If the current descriptor processes the last segment of the IV then total IV length should be provided, otherwise total length of textdata should be provided.
9–10	Initial counter value Y_0. Normally, this value is a result of the IV stream processing and needs to be provided only if the message is split into multiple descriptors and for those descriptors that come after IV processing is complete. Otherwise, value provided here is ignored and overwritten with computed Y_0. In case of GCM-GHASH cipher mode setting, constant H from GHASH(H, AAD, ciphertext) should be provided in these registers. Note that this, in the general case, does not have to be equal to E(K, $\{0\}^{128}$) where K is a key as defined for GCM.
11	Length of AAD in bits. This pertains to the length of the AAD part processed in the current descriptor. If the current descriptor does not process AAD, 0 should be written. If AAD is not split into multiple descriptors, this field should contain the total AAD length. The value written here should be divisible by 128 for all AAD segments except for the last one that can be any number of bits. Note, however, that the actual AAD stream supplied to the AES engine through FIFOs has to be zero-padded to an integral number of 16-byte blocks.
12	Length of IV in bits or a part of it processed in the current descriptor. Similar remarks apply like in the case of AAD. This register is not used for GCM-GHASH.

Table 27-7 to **Table 27-10** describe the proper usage of context registers in case of encryption, decryption, GCM w/ICV and GCM-GHASH cipher mode settings. The context is in each case described in terms of the input context required for starting new GCM processing or continuation of processing after context switch, and in terms of the results (output) stored in context registers after GCM execution run is completed.

Table 27-7. GCM Encryption Context

Context Register	GCM Encrypt (outbound)				
	Mode Register (ECM = 10, AUX0 = 0, CM = 01, ED = 1)				
	AUX1 Value			AUX2 Value	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	MAC (Computed)
2					
3	—			—	
4					
5	Y_i (Counter)				
6					
7	—	len(AAD)^{T*}	—		
8	—	len(textdata)^{T*}	len(IV)^{T}		
9	Y_0 (Initial Counter)				
10					
11	len(AAD)^{C*}				
12	len(IV)^{C*}				

Notes:
1. * = Must be written at the start of a new message, except if zero
2. ^{C} = length of data processed with current descriptor (in bits)
3. ^{T} = length of total data (in bits)
4. — = ignored

Table 27-8. GCM Decryption Context

Context Register	GCM Decrypt (inbound)				
	Mode Register (ECM = 10, AUX0 = 0 or 1, CM = 01, ED = 0)				
	AUX1 Value			AUX2 Value	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)				
2					
3	—			—	MAC (Computed)
4					
5	Yi (Counter)				
6					
7	—	$len(AAD)^{T*}$	—		
8	—	$len(textdata)^{T*}$	$len(IV)^{T*}$		
9	Y_0 (Initial Counter)				
10					
11	$len(AAD)^{C*}$				
12	$len(IV)^{C*}$				

Notes:
1. * = Must be written at the start of a new message, except if zero
2. C = length of data processed with current descriptor (in bits)
3. T = length of total data (in bits)
4. — = ignored

Table 27-9. GCM w/ICV Context

Context Register	GCM w/ICV (inbound)				
	Mode Register (ECM = 11, AUX0 = 0 or 1, CM = 01, ED = 0)				
	AUX1 Value			AUX2 Value	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	MAC (Computed and truncated to icv_size most significant bytes)
2					
3	MAC (Received)				
4					
5	Y_i (Counter)				
6					
7	—	len(AAD)^{T*}	—		
8	—	len(textdata)^{T*}	len(IV)^{T}		
9	Y_0 (Initial Counter)				
10					
11	len(AAD)^{C*}				
12	len(IV)^{C*}				

Notes:
1. * = Must be written at the start of a new message, except if zero
2. ^{C} = length of data processed with current descriptor (in bits)
3. ^{T} = length of total data (in bits)
4. — = don't care

Table 27-10. GCM-GHASH Context

Context Register	GCM-GHASH (only GHASH computed)				
	Mode Register (ECM = 10, AUX0 = 1, CM = 01, ED = 1)				
	AUX1 Value			AUX2 Value	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	MAC (Computed)
2					
3					
4					
5					
6					
7	—	len(AAD)^{T*}	—		
8	—	len(textdata)^{T*}	len(IV)^{T}		
9	H*				
10					
11	len(AAD)^{C}				
12	—				

Notes:
1. * = Must be written at the start of a new message, except if zero
2. ^{C} = length of data processed with current descriptor (in bits)
3. ^{T} = length of total data (in bits)
4. — = don't care

As an example, lets consider the case of GCM encrypt operation that generates the final MAC tag and where the whole message is small enough that it can be processed with one descriptor. The mode bits should be configured as ECM = 10, AUX[2–0] =100, CM = 01, and ED = 1. Only context registers 11–12 need to be written with the bit lengths of AAD and IV, respectively. The bit length of textdata should be written to the data size register up to the size of 2^{19} bits. IV, AAD, and textdata in that order should be sent through the input FIFO and the result (textdata) read from the output FIFO as available. At the end, the final MAC tag is read from the context registers 1–2.

27.6.3.10 AESU Key Registers

The AESU key registers hold from 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to key 1U. Any key data written to bytes beyond the value written to the Key Size Register is ignored. The Key Data Registers are cleared when the AESU is reset or reinitialized. If these registers are modified during message processing, a context error is generated.

27.6.3.11 AESU FIFOs

AESU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the AESU FIFO address space enqueues data to the AESU input FIFO, and a read from anywhere in the AESU FIFO address space dequeues data from the AESU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the staging register are written, the entire 8 bytes is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the AESU End_of_message register is written.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, that 8 bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflows caused by reading or writing the AESU FIFOs are reflected in the AESU Interrupt Status Register.

The AESU fetches data 128 bits at a time from the input FIFO. During processing, the input data is encrypted or decrypted with the key and initialization vector (CBC mode only) and the results are placed in the output FIFO. The output size is the same as the input size.

The input FIFO can be written any time the number of 8-byte sets currently in the input FIFO (as indicated by the IFL field of the AESU Status Register) is less than 32. There is no limit on the total number of bytes in a message. The number of bits in the final message block must be set in the Data Size Register.

The output FIFO can be read any time the OFR signal is asserted (as indicated in the AESU Status Register). This indicates that the number of bytes in the output FIFO is at or above the threshold specified in the Mode Register.

For AES-CCM mode, the input data stream to the input FIFO consists of the CCM Header, followed by the Additional Authenticated Data (AAD), followed by Plaintext if encrypting, or

Ciphertext if decrypting. This mode does not support zero length AAD and zero length payload. Note that AESU only supports 2 byte CCM headers and does not support extended CCM headers.

27.6.4 Message Digest Execution Unit (MDEU)

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using the MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 algorithms for bulk data hashing.

With any hash algorithm, the larger message is mapped onto a smaller output digest, so it is possible for two messages to produce the same digest. The hash digest lengths (128 bits or more) are sufficiently large, however, that such collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally unfeasible to construct two distinct but similar messages that produce the same hash output. The MDEU is designed to support the following cryptographic algorithms:

- The MD5 generates a 128-bit hash, and the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the NIST FIPS 180-1 standard.
- SHA-224 is a 224-bit hash function, specified by the NIST FIPS 180-2 standard.
- SHA-256 is a 256-bit hash function that provides 256 bits of security against collision attacks, specified by the NIST FIPS 180-2 standard.
- SHA-384 is a 384-bit hash function, specified by the NIST FIPS 180-2 standard.
- SHA-512 is a 512-bit hash function, specified by the NIST FIPS 180-2 standard.
- The MDEU also supports HMAC computations, as specified by the NIST FIPS-198 standard.

If a digest is supplied to the MDEU, it can do a bitwise check of this supplied digest against the one computed by the MDEU (ICV checking). In a typical operation, the MDEU is used through channel-controlled access, which means that most reads and writes of MDEU registers are directed by the SEC channels. Driver software performs core processor-controlled register accesses only on a few registers for initial configuration and error handling.

The following subsections include a description of ICV checking functionality and general descriptions of the MDEU registers and structures. **Section 27.7**, *Programming Model*, on page 27-95 provides a detailed description of each register and associated register fields.

27.6.4.1 ICV Checking

This EU includes an ICV checking feature, that is, it can generate an ICV and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the core processor either via interrupt by a writeback of EU status fields into core processor memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-196), and mask the ICE bit in the Interrupt Mask Register (**Section 27.6.4.8**, *MDEU Interrupt Mask Register*, on page 27-71). In this case the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, there is an error interrupt to the core processor, but no done interrupt or writeback.

27.6.4.2 MDEU Mode Register

The MDEU Mode Register is used to program the function of the MDEU. For normal operation through a channel, the least significant byte of this register is specified through the MODE0 or MODE1 field of the descriptor header. The remaining bits are supplied by the channel and thus are not under direct user control.

The MDEU Mode Register has two bits used to define the register configurations:

- *MDEU_B bit.* Determines which of two sets of algorithms are available through the ALG bits. The two sets of algorithms are:
  - MDEU-A set (MD5, SHA-1, SHA-224, and SHA-256) selected when MDEU_B = 0.
  - MDEU-B set (SHA-224, SHA-256, SHA-384, and SHA-512) selected when MDEU_B = 1

  In channel-controlled operation, the channel supplies the MDEU_B mode bit value based on the EU_SEL field of the descriptor header:
  - MDEU-A is selected if EU_SEL = 0b0011.
  - MDEU-B is selected if EU_SEL = 0b1011

  See **Section 27.7.1.2**, *Descriptor Header*, on page 27-100 for details.
- *NEW bit*. Determines the configuration of the other mode register fields (see **Section 27.7.10.1**, *MDEU Mode Register (MDEUMR)*, on page 27-248 for alternate register fields). The new configuration (NEW=1) is used only by TLS/SSL descriptor types (1000_1, 1001_1). The old configuration (NEW=0) is used by all other descriptor types and is the almost the same as defined in the SEC 2.0 specification, except for the CICV and SMAC bits. When MDEU is configured by the Polychannel, the value of NEW is determined by the descriptor type field of the descriptor header. If the descriptor type field indicates an SSL/TLS type descriptor, the Polychannel sets the NEW bit; otherwise NEW is cleared.

The Mode Register is cleared when the MDEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated.

The most common task likely to be executed via the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPSec, and TLS. The SSL 3.0 protocol uses a slightly different SSL-MAC. If an HMAC or SSL-MAC is to be performed using a single descriptor (with the MDEU acting as sole or secondary EU), use the Mode Register bit settings listed in **Table 27-11**

Table 27-11. Mode Register—HMAC or SSL-MAC Generated by Single Descriptor

Bits	Field	Value	
		for HMAC	for SSL-MAC
56	CONT	0 (off)	0 (off)
58	SMAC	0(on)	1(on)
59	INIT	1(on)	1(on)
60	HMAC	1(on)	0(on)

To generate an HMAC for a message that is spread across a sequence of descriptors, use the Mode Register bit settings listed in **Table 27-12**.

Table 27-12. Mode Register—HMAC Generated Across a Sequence of Descriptors

Bits	Field	Value		
		First Descriptor	Middle Descriptor(s)	Final Descriptor
56	CONT	1 (on)	1 (on)	0 (off)
59	INIT	1 (on)	0 (off)	0 (off)
60	HMAC	1 (on)	0 (off)	1 (on)

All descriptors other than the final descriptor must output the intermediate message digest for the next descriptor to reload as MDEU context. SSL-MAC operations cannot be spread across a sequence of descriptors. Additional information on descriptors can be found in **Section 27.7.1.1**, *Descriptor Structure*, on page 27-99.

27.6.4.3 MDEU Key Size Register

The MDEU key size value indicates the number of bytes of key memory that should be used in HMAC generation. MDEU supports at one key block. MDEU generates a key size error if the value written to this register exceeds 64 bytes for MD5, SHA-1, SHA-224, or SHA-256, or if the value exceeds 128 bytes for SHA-384 or SHA-512.

27.6.4.4 MDEU Data Size Register

The MDEU Data Size Register indicates the number of bits of data to be processed. The Data Size field is a 21-bit signed number. Values written to this register are added to the current

register value. Multiple writes are allowed. The MDEU processes data when there is a positive value in this register and there is data available in the MDEU input FIFO. (Negative values can arise in inbound processing, when it is necessary to hold back data from the MDEU until the pad length is decrypted.). Because the MDEU does not support bit offsets, the least significant 3 bits must be written as 0 and are always read as zero. Furthermore, when the CONT bit of the MDEU Mode Register is high, the data size must be a multiple of the block size (that is, 512 bits for MD5, SHA-1, SHA-224, and SHA-256; 1024 bits for SHA-384 and SHA-512). Violating either of these conditions causes a data size error (the MDEUIDR[DSE] bit is set).

This register is cleared when the MDEU is reset or reinitialized. At the end of processing, its contents is decremented down to zero (unless there is an error interrupt).

Note: Writing to the Data Size Register allows the MDEU to enter auto-start mode. Therefore, the required Context Registers must be written prior to writing the data size.

27.6.4.5 MDEU Reset Control Register

This register allows three levels of reset just for the MDEU, as defined by the three self-clearing bits.

27.6.4.6 MDEU Status Register

This Status Register reflects the state of the MDEU internal signals. The majority of these internal signals reflect the state of low-level MDEU functions, such as data padding, key padding, and so forth, and are not important to the user, however the user should be aware that reads of this register, especially during processing, are likely to return non-zero values for many of the most significant bits. The four least-significant bit fields are most likely to be of interest to the user.

The MDEU Status Register is read-only. Writing to this location result in an address error being reflected in the MDEU Interrupt Status Register.

27.6.4.7 MDEU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the MDEU Interrupt Mask Register is zero (see **Section 27.7.10.7**, *MDEU Interrupt Mask Register (MDEUIMR)*, on page 27-256).

If the MDEU Interrupt Status Register is non-zero, the MDEU halts and the MDEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186). In addition, if the MDEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel*

Status Registers (CSR[1–4]), on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit in the MDEU Reset Control Register.

27.6.4.8 MDEU Interrupt Mask Register

The MDEU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.7.10.6**, *MDEU Interrupt Status Register (MDEUISR)*, on page 27-254), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.4.9 MDEU ICV Size Register

The MDEU ICV size register stores the number of bytes of the ICV result to compare if the MDEU performs ICV comparison (see **Section 27.7.10.1**, *MDEU Mode Register (MDEUMR)*, on page 27-248). This register is cleared when the MDEU is reset or reinitialized.

27.6.4.10 MDEU End_of_Message Register

The End_of_Message Register in the MDEU indicates that an authentication operation is completed. After the final message block is written to the input FIFO, the End_of_Message Register must be written. The value in the Data Size Register is used to determine how many bits of the final message block (512) are processed. Note that this register has no data size, and during the write operation, the core processor data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to initiate a done interrupt.

27.6.4.11 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provides the resulting message digest or HMAC, along with an aggregate bit count.

Note: All SHA algorithms are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU Mode Register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In

this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location before the module is done generates an error interrupt.

After a power-on reset, all the MDEU Context Register values are cleared (0). The MDEU Context Registers are initialized if the INIT bit is set in the MDEU Mode Register. However, all registers are initialized, regardless of mode selected, but only the appropriate Context Register values are used in hash generation per the mode selected. Even though the user typically does not need to know about the MDEU Context Register initialization values; they are documented for completeness in the event that the user reads these registers using core processor-controlled access. MDEU resets via the MDEU Reset Control Register (**Section 27.6.4.5**, *MDEU Reset Control Register*, on page 27-70) or SEC global software reset (**Section 27.7.4.1**, *Master Control Register (MCR)*, on page 27-177) do not clear these registers.

27.6.4.12 MDEU Key Registers

The MDEU maintains sixteen 64-bit registers for writing an HMAC key. Only the first eight registers are used for MD-5, SHA-1, SHA-224, or SHA-256. The IPAD and OPAD operations are performed automatically on the key data when required.

Note: All SHA algorithms are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU Mode Register indicates MD5 is the hash of choice.

27.6.4.13 MDEU FIFOs

MDEU uses an input FIFO to hold data to be hashed (followed in some case by an ICV value for ICV checking). Normally, the channels control all access to this FIFO. For core processor-controlled operation, a write to anywhere in the MDEU FIFO address space enqueues data to the MDEU input FIFO, and a read from anywhere in this address space returns all zeros.

When the core processor writes to the MDEU FIFO (using core processor-controlled access), it can write to any FIFO address using byte, 4-byte, or 8-byte accesses. The MDEU assembles these bytes from left to right, that is, the first bytes written are placed in the most significant bit-positions. Whenever the MDEU accumulates 8 bytes, these 8 bytes are automatically enqueued into the FIFO, and any remaining bytes are left-justified in preparation for assembling the next 8 bytes. It is not necessary to fill all bytes of the final 8-byte set. Any remaining bytes in the staging register are automatically padded with zeros and forced into the input FIFO when the MDEU End_of_Message Register is written.

Overflows caused by writing the MDEU FIFO are reflected in the MDEU Interrupt Status Register.

27.6.5 ARC Four Execution Unit (AFEU)

In typical operation, the AFEU is used through channel-controlled access, which means that most reads and writes of AFEU registers are directed by the SEC channels. Driver software would perform core processor-controlled register accesses only on a few registers for initial configuration and error handling.

The following subsections include general descriptions of the AFEU registers and structures. **Section 27.7**, *Programming Model,* on page 27-95 provides a detailed description of each register and associated register fields.

27.6.5.1 AFEU Mode Register

The AFEU Mode Register contains three bits that are used to program the AFEU. The Mode Register is cleared when the AFEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated.

27.6.5.1.1 Core Processor-Provided Context via Prevent Permute

In the default mode of operation, the core processor provides the key and key size to the AFEU. The initial memory values in the S-Box are permuted with the key to create new S-Box values, which are used to encrypt the plaintext.

If the ‘Prevent Permute’ mode bit is set, the AFEU does not require a key. Rather, the core processor writes the context to the AFEU and message processing occurs using the provided context. This mode is used to resume processing of a message using the already permuted S-Box. The context can be written through the FIFO if the ‘context source’ mode bit is set. The AFEU context is 259 bytes long, and must be in the format provided by the Dump Context function (see Section 27.6.5.1.2, **Dump Context**).

27.6.5.1.2 Dump Context

This mode can be independently specified in addition to core processor-provided context mode. In this mode, once message processing is complete and the output data is read, the AFEU makes the current context data available for reads via the output FIFO. The AFEU context is 259 bytes long.

Note: After the initial key permute to generate a context for an AFEU encrypted session, all subsequent messages reuses that context, such that it is loaded, modified during the encryption, and unloaded, similar to the use of a CBC initialization vector in DES operations. A new context is generated (via key permute) according to a re-keying interval specified by the security protocol. Context should never be loaded to encrypt a message if a key is loaded and permuted at the same time.

27.6.5.2 AFEU Key Size Register

The AFEU key size value indicates the number of bytes of key memory to use in performing S-box permutation. Any key data beyond the number of bytes in the Key Size Register is ignored. This register is cleared when the AFEU is reset or reinitialized. If the key size specified is less than 1 or greater than 16, a key size error is generated. If the Key Size Register is modified during processing, a context error is generated. Note: Although the AFEU supports key lengths as short as 1 byte, a 1-byte key offers little security. Most ARC-4 applications specify keys of 5–16 bytes.

Note: The device driver creates properly formatted descriptors for situations requiring a key permute prior to ciphering. When using core processor-controlled access (typically for debug), the user must set the AFEU Mode Register to perform ‘permute with key’, then write the key data to AFEU key registers, then write the key size to the Key Size Register. The AFEU starts permuting the memory with the contents of the key registers immediately after the key size is written.

27.6.5.3 AFEU Context/Data Size Register

The AFEU context/Data Size Register stores the number of bits in the final message with an upper bound of 4096. Whatever number is written (and whatever truncated value is stored) must be a multiple of 8. This value controls how much data is processed from the last block. The last message block must be a multiple of 8 from 8–64. If you write data size that is not a multiple of 8, a data size error is generated. Only the 3 lsbs are checked to determine if there is a data size error. Because all upper bits are ignored, the entire message length (in bits) can be written to this register.

The context/Data Size Register is also used to specify the context size, when context is used. The context size is fixed at 2072 bits (259 bytes). When loading context through the FIFO, all context data must be written prior to writing the context data size. The message data size must be written separately.

Note: When reloading an existing context using core processor-controlled access, the user must write the context to the input FIFO, then write the context size (always 2072 bits). The write of the context size indicates to the AFEU that all context is loaded. The user then writes the message data size to the context/Data Size Register. After this write, the user can begin writing message data to the FIFO.

Writing to this register causes the AFEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

This register is cleared when the AFEU is reset or reinitialized.

27.6.5.4 AFEU Reset Control Register

This register allows 3 levels of reset that effect the AFEU only, as defined by 3 self-clearing bits. The AFEU executes an internal reset sequence for hardware reset, software reset, or module initialization, which performs proper initialization of the S-Box. Use the RESET_DONE bit in the AFEU Status Register to determine when the reinitialization is complete.

27.6.5.5 AFEU Status Register

This Status Register reflects the state of AFEU internal signals. The AFEU Status Register is read-only. Writing to this location results in address error being reflected in the AFEU Interrupt Status Register.

27.6.5.6 AFEU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the AFEU Interrupt Mask Register is zero (see **Section 27.7.11.7**, *AFEU Interrupt Mask Register (AFEUIMR)*, on page 27-270).

If the AFEU Interrupt Status Register is non-zero, the AFEU halts and the AFEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186). In addition, if the AFEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the AFEU Reset Control Register.

27.6.5.7 AFEU Interrupt Mask Register

The Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.7.11.6**, *AFEU Interrupt Status Register (AFEUISR)*, on page 27-268), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.5.8 AFEU End_of_Message Register

The End_of_Message Register in the AFEU is used to signal the AFEU that all processed data is written to the input FIFO. This allows the AFEU to do special processing when it reaches the last

block of data. Before this register is written, the AFEU does not process the last block of data in its input FIFO. After this register is written, the AFEU continues to do normal processing on all except the last block of data, and then goes on to process the last block using the value in the Data Size Register to determine how much of the block to process. The Data Size Register specifies the number of bits to process, which must be a multiple of 8 from 8–64. Once processing of the last block is completed, the AFEU issues a done interrupt. If the dump context bit in the AFEU Mode Register is set, the context is written to the output FIFO after the last message. A read of the AFEU End_of_Message Register always returns a zero value.

27.6.5.9 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

27.6.5.9.1 AFEU Context Memory

The S-Box memory consists of 256 bytes of SRAM, each readable and writable as part of a 64-bit set. Do not write the S-Box contents with data unless that data is previously read from the S-Box. Context data should only be written if the ‘prevent permutation’ mode bit in the AFEU Mode Register is set (see **Section 27.6.5.1**, *AFEU Mode Register*, on page 27-73). After context data, the context length must be written to the context/data length register (see **Section 27.7.11.3**, *AFEU Context/Data Size Register (AFEUCDSR)*, on page 27-265). After this, message data can be written. If the Context Registers are written during message processing or the ‘prevent permutation’ bit is not set, a context error is generated.

Valid context data can only be read after processing is done. Reading context data before the module is done generates an error interrupt.

Context data can be written and read either via the Context Registers or via the input and output FIFOs. The user specifies which through the CS bit of the AFEU Mode Register (see Section 27.6.5.1, **AFEU Mode Register**). See Section 27.6.5.11, **AFEU FIFOs** for more about addressing the FIFOs.

27.6.5.9.2 AFEU Context Memory Pointer Register

The context memory pointer register holds the internal context pointers that are updated with each byte of message processed. These pointers correspond to the values of I, J, and Sbox[I+1] in the ARC-4 algorithm. If this register is written during message processing, a context error is generated.

When performing ARC-4 operations, the user has the option of performing a new S-Box permutation per packet, or unloading the contents of the S-box (context) and reloading this context prior to processing of the next packet. The S-Box contents (256 bytes) plus the three bytes of the context memory pointers are unloaded and reloaded via the AFEU FIFOs.

AFEU Context consists of the contents of the S-Box, as well as three counter values, which indicate the next values to be used from the S-Box. Context must be loaded in the same order in which it was unloaded.

27.6.5.10 AFEU Key Registers

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-Box, in conjunction with the AFEU Key Size Register. AFEU performs permutation starting with the first byte of key register 0, and uses as many bytes from the two key registers as necessary to complete the permutation. Reading either of these memory locations generates an address error interrupt.

27.6.5.11 AFEU FIFOs

AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the AFEU FIFO address space enqueues data to the AFEU input FIFO, and a read from anywhere in the AFEU FIFO address space dequeues data from the AFEU output FIFO.

In the special case where context is written through the input FIFO, the first write must be to range address 3_8E00 – 3_8E07. Context reads can be from any address in the FIFO address range.

Writes to the input FIFO go first to a staging register which can be written by byte, 4-byte, or 8-byte accesses). When all 8 bytes of the staging register are written, the entire 8 bytes is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the AFEU End_of_Message Register is written.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, that 8 bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflow caused by reading or writing the AFEU FIFOs are reflected in the AFEU Interrupt Status Register.

27.6.6 Kasumi Execution Unit (KEU)

The Kasumi execution unit (KEU) is designed to support the F8 confidentiality function of the 3GPP, GSM A5/3, EDGE A5/3 and GPRS GEA3 algorithms and also the 3GPP F9 integrity function. This section contains details about the Kasumi Execution Unit (KEU), including modes of operation, status and control registers, and FIFOs.

In typical operation, the KEU is used through channel-controlled access, which means that most reads and writes of KEU registers are directed by the SEC channels. Driver software would perform core processor-controlled register accesses only on a few registers for initial configuration and error handling.

This EU includes an ICV checking feature, that is, it can generate an ICV and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the core processor either via interrupt by a writeback of EU status fields into core processor memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see **Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4])*, on page 27-196), and mask the ICE bit in the Interrupt Mask Register (**Section 27.7.12.7**, *KEU Interrupt Mask Register (KEUIMR)*, on page 27-282). In this case the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, e an error interrupt is generated to the core processor, but no done interrupt or writeback is issued.

The following subsections include general descriptions of the KEU registers and structures. **Section 27.7**, *Programming Model*, on page 27-95 provides a detailed description of each register and associated register fields.

27.6.6.1 KEU Mode Register

The KEU Mode Register contains several bits used to program the KEU. The Mode Register is cleared when the KEU is reset or reinitialized. Setting a reserved mode bit generates a data error. Setting both the GSM and EDGE bits to one generates a data error. If the KEU Mode Register is modified during processing, a context error is generated.

27.6.6.2 KEU Key Size Register

The KEU Key Size Register stores the number of bytes in the key. It should be set to 16 bytes. This register is cleared when the KEU is reset or reinitialized. If a specified key size does not match the selected algorithm(s), an illegal key size error is generated.

27.6.6.3 KEU Data Size Register

The KEU Data Size Register stores the number of bits to process in the final message set. Because Kasumi allows for bit level granularity for encryption/decryption, there are no illegal data sizes. The user must write the proper bit length of the message to notify the KEU of any padding performed by the core processor. This register is cleared when the KEU is reset or

reinitialized. Writing to this register signals the KEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

Example 27-1. F8 Function Example

If the 64-bit F8 keystream is ‘0x1234567890abcdef’and the Data Size Register contains 0x0a (10 = 1 byte + 2 bits), the final ten message bits is XORed with ten bits of keystream ‘0x120’. The PE (Process End of Message) mode bit must be set (see **Section 27.7.12.1**, *KEU Mode Register (KEUMR)*, on page 27-274). If PE is cleared (= 0), processing can not occur.

Example 27-2. 3GPP F9 Function Example

The final 64 bits of the message are padded as specified in the 3GPP F9 algorithm. The Process End of Message (PE) mode bit must be set. 3GPP F9 padding is automatically performed. The Communication Direction (CD) bits and ‘1’ are appended to the end of the message. The result is then zero-padded to 64 bits.
For example, if the last block is xF81A000000000000 (big endian) and data size contains 0x0F (15 bits = 1 byte + 7 bits), the most-significant 15 bits (underlined) of the last block (0xF81A = 1111_1000_0001_101) are padded left to right as follows:
1111_1000_0001_101$_1000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000
where ‘$’ is the value of the CD bits in the Mode Register. If PE is cleared (= 0), processing can not occur.

27.6.6.4 KEU Reset Control Register

The KEU Reset Control Register allows 3 levels reset of the KEU, as defined by the 3 self-clearing bits:

27.6.6.5 KEU Status Register

The KEU Status Register is a read-only register that reflects the state of six status outputs. Writing to this location results in an address error being reflected in the KEU Interrupt Status Register.

27.6.6.6 KEU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the KEU Interrupt Mask Register is zero (see **Section 27.7.12.7**, *KEU Interrupt Mask Register (KEUIMR)*, on page 27-282).

If the KEU Interrupt Status Register is non-zero, the KEU halts and the KEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186). In addition, if the KEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU

error bit is set in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit in the KEU Reset Control Register.

27.6.6.7 KEU Interrupt Mask Register

The KEU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.7.12.6**, *KEU Interrupt Status Register (KEUISR)*, on page 27-280), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the KEU Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the KEU Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.6.8 KEU Data Out Register (F9 MAC)

Following a done interrupt, the read-only KEU data out register holds the F9 message authentication code. A 64-bit value is returned. This value can be truncated to 32 bits for some applications. Writing to this location results in an address error being reflected in the KEU Interrupt Status Register.

Note: According to the ETSI/SAGE 3GPP specification for F9 (version 1.2), only 32 bits of the final MAC are used. This is the lower 4 bytes of the KEU data out register.

27.6.6.9 KEU End_of_Message Register

The KEU End_of_Message Register signals the KEU that the final message block is written to the input FIFO. Writing to this register causes the KEU to process the final block of a message, allowing it to signal a done interrupt. When processing the last block, the value in the Data Size Register determines how many bits of the final message set (1–64) are processed. The value written to this register does not matter. A read of this register always returns a zero value.

27.6.6.10 KEU IV_1 Register

The KEU IV_1 register is a general purpose IV register used during the initialization phase of the F8 algorithms for 3GPP, GSM A5/3, EDGE A5/3, GPRS GEA3 and also for the F9 algorithm for 3GPP. The user must write the appropriate value as defined by the standards for each algorithm before a new message is started. Once the initialization phase is completed, the KEU IV_1 register is no longer used for the remainder of F8 or F9 processing. However, if 3GPP F9, is selected, because the KEU IV_1 register contains the direction bit as defined by the 3GPP standard, the KEU IV_1 register MUST be written back during context switches, to complete the generation of the 3GPP MAC.

Note: The user must ensure that fields of the IV_1 register are programmed correctly in accordance with the selected algorithm.

27.6.6.11 KEU ICV_In Register

If ICV checking is required, then the value to compare with the computed F9 MAC value must be written to the ICV_In register before data size is written.

27.6.6.12 KEU IV_2 Register (Fresh)

The fresh value in the KEU IV_2 register is used during the initialization phase of the 3GPP F9 algorithm. This value is ignored when the F8 algorithm is selected. The fresh value must be written before a new message to be processed with 3GPP F9 is started. Once the initialization phase is completed, the KEU IV_2 register is no longer used during message processing. You do not need to write the KEU IV_2 register during context switches.

27.6.6.13 KEU Context Data Registers

There are six 64-bit KEU context data registers that allow the core processor to read/write the contents of the context used to process the message. The KEU context data registers must be read when changing context and restored to their original values to resume processing an interrupted message. For F8 and 3GPP F9 modes, all six 64-bit KEU context data registers must be read to retrieve context, and all 6 must be written back to restore context. The context must be written prior to the key data. If the any of the KEU context data registers are written during message processing, a context error is generated. All KEU context data registers are cleared when a hard/soft reset or initialization is performed.

Note: In typical operation, a frame is received and processed in its entirety, with the KEU performing session specific initialization using the contexts of KEU IV_1 and IV_2 registers. The KEU context data and IV_1 registers should only be unloaded/reloaded when the processing of a frame is discontinued prior to completion, then processing is resumed.

27.6.6.14 KEU Key Data Registers_[1–2] (Confidentiality Key)

The first pair of KEU Key data registers together hold one 128-bit key used for F8 encryption/ decryption. KEU Key Data Register_1, (CK-high), holds the first 8 bytes (1–8). KEU Key Data Register_2, (CK-low), holds the second 8 bytes (9–16). The KEU Key Data Registers must be written before message processing begins. Writing a register while the block is processing data, generates a context error. Reading from either of these registers results in an address error being reflected in the KEU Interrupt Status Register.

27.6.6.15 KEU Key Data Registers _[3–4] (Integrity Key)

The second pair of KEU Key Data Registers together hold one 128-bit key that is used for F9 message authentication. KEU Key Data Register_3, (IK-high) holds the first 8 bytes (1–8). KEU Key Data Register_4, (IK-low), holds the second 8 bytes (9–16). The KEU Key Data Registers must be written before message processing begins. Writing a register while the block is processing data generates a context error.

If the ‘F9 only’ mode is set, the integrity key data can optionally be written to KEU Key Data Registers_[1–2]. This eliminates the need for the core processor to offset from the base key address to write to KEU Key Data Registers[3–4] while using the KEU exclusively for the F9 integrity function.

Reading from either of these registers results in an address error being reflected in the KEU Interrupt Status Register.

27.6.6.16 KEU FIFOs

KEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the KEU FIFO address space enqueues data to the KEU input FIFO, and a read from anywhere in the KEU FIFO address space dequeues data from the KEU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, 4 bytes, or 8 bytes. When all 8 bytes of the staging register are written, the entire 8 bytes are automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the KEU End_of_Message Register is written.

The output FIFO is readable in byte, 4-byte, or 8-byte increments. When all 8 bytes of the header are read, that 8 bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU. Overflows and underflows caused by reading or writing the KEU FIFOs are reflected in the KEU Interrupt Status Register.

The KEU fetches data 64 bits at a time from the KEU Input FIFO. During F8 processing, the input data is XORed with the generated keystream and the results are placed in the KEU Output FIFO. During F9 processing, the input data is hashed with the integrity key and the resulting MAC is placed in the KEU data out register. The output size is the same as the input size.

27.6.7 Cyclic Redundancy Check Unit (CRCU)

The CRCU computes a single 32-bit cyclic redundancy code (checksum) from all data presented on the input bus. The CRC algorithm treats a message stream of bits as coefficients of a massive polynomial and computes the remainder of the modulo two division by an order 32 divisor polynomial. The divisor polynomial is specific to the protocol and chosen to conform to certain mathematical properties to ensure that single bit errors can be detected. Cyclic redundancy codes are used to ensure data integrity over potentially unreliable channels. There are two major CRC protocol algorithms: CRC32 and CRC32C. **IEEE** Std. 802 defines the CRC32 algorithm, while iSCSI defines the CRC32C algorithm. Both protocols bit swap, byte swap, and then complement the calculated remainder to generate the checksum. The CRCU is designed to support the following check algorithms:

- CRC32 algorithm is specified in **IEEE** Std. 802.1.
- CRC32C algorithm is specified in RFC3385.
- Programmable polynomial mode with optional remainder bit mangling. This can be used to implement proprietary protocols.

The CRCU can perform ICV checking by computing a raw CRC across a message and previously-calculated CRC. Integrity is verified if the result matches the polynomial specific residue.

The following sections describe the CRCU modes of operation, status and control registers, and FIFOs. Most of the registers are not normally accessed by the core, but are documented primarily for debug purposes. In typical operation, the CRCU is used through channel-controlled access, which means that most reads and writes of CRCU registers are directed by the SEC channels. Driver software perform core-controlled register accesses only on a few registers for initial configuration and error handling. Detailed programming information is provided in **Section 27.7**.

27.6.7.1 ICV Checking

This EU includes an ICV checking feature, that is, it can verify a message/CRC pair by calculating a raw CRC and comparing it to the polynomial specific residue. The pass/fail result of this check can be returned to the core either via interrupt by a writeback of EU status fields into core memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see **Section 27.7.6.1**), and mask the CICV Fail bit in the CRCU Interrupt/Error Mask Register (**Section 27.7.13.8**). For this case, the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the CICVF bit in the CRCU Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no CRC mismatch, then the normal done signalling (by interrupt or writeback) occurs.

When there is a CRC mismatch, an error interrupt is sent to the core, but no done interrupt or writeback.

27.6.7.2 CRCU Mode Register

The CRCU Mode Register programs the function of the CRCU and is generally the first register written. A context error is generated if this register is written after processing begins.

27.6.7.3 CRCU Key Size Register

The Key Size Register stores the number of valid bytes in the dynamic polynomial written to the Key Register. This value is reset to zero, and if a value other than zero, four, or eight is written to it, an illegal key size error is generated. It is not necessary to write to this register, because the polynomial size is clearly fixed by the algorithm. A context error is generated if this register is written after processing begins.

27.6.7.4 CRCU Data Size Register

The Data Size Register is written with the number of bits of data to process. Writing to this register puts the CRCU module into a busy state and starts data processing. This register can be written multiple times while data processing is in progress. The actual values written are ignored, although an error is generated if the value is not a multiple of 8 bits.

27.6.7.5 CRCU Reset Control Register

The Reset Control Register controls the reset/re-initialization of the block.

27.6.7.6 CRCU Control Register

The Control Register stores the static polynomial and residue used in custom CRC computations. The reset value of this register is the **IEEE** 802 standard CRC32 residue coefficient “r” and polynomial coefficient “g”. This register is static, in that it is only reset by performing a software reset, not by an EU reinitialization. This allows an application specific custom polynomial to be written to the register once and used many times. A context error is generated if this register is written after processing begins. A polynomial error is generated if a value is written to this register that does not have a one in g^0.

27.6.7.7 CRCU Status Register

The Status Register provides general information about the status of the CRCU. A read of the status register captures a snapshot of CRCU operating state at a particular moment in time. The three interrupt flags (error, done, and reset), provide visibility to the respective EU ports used to notify the core about the operation status. Writes to this register are ignored.

27.6.7.8 CRCU Interrupt/Error Status Register

The Interrupt/Error Status Register latches the source of various error interrupts. This register is both readable and writable, allowing the setting and/or clearing of the individual status bits. All status bits are maskable by setting the corresponding bit in the Interrupt/Error Mask Register. A status bit must be unmasked for it to be set. Individual status bits can be cleared by writing a zero to the respective bits. If these bits are masked, writing a one still clears them. The entire Interrupt/Error Status Register can be cleared, except for the Halt on Error bit, by resetting/reinitializing CRCU or by writing to the Clear IRQ bit in the Reset Control Register. The Halt on Error bit can only be cleared by resetting/reinitializing the CRCU. If an error occurs and is not masked, the error interrupt signal is asserted and the CRCU stops processing. If the error is masked, the interrupt is not asserted and the CRCU continues processing with likely bizarre results.

27.6.7.9 CRCU Interrupt/Error Mask Register

The Interrupt/Error Mask Register allows for individual error sources to be masked, thus preventing the error interrupt signal from being asserted. Error status is not captured for any error bits that are masked. Any core writes to a masked error bit result in clearing that error bit. Masking an error bit allows for a hardware error condition to occur potentially undetected. Therefore, use extreme care when masking errors because it can produce invalid results. Do not mask errors except for debug operation. This register can be reset by resetting the CRCU.

27.6.7.10 CRCU ICV Size Register

The ICV Size Register is 64-bit readable/writable for compatibility with the MDEU. Values written to this register are always ignored and a read of this register always returns zero. A context error is generated if this register is written after processing begins.

27.6.7.11 CRCU End of Message Register

The End of Message Register is used to indicate that all data is written to the CRCU. A write to this register is required to complete a CRC32 operation. The CRCU starts processing message data as soon as the Data Size Register is written and data becomes available in the FIFO, but it does not process a remaining partial word or perform an ICV check until a write to this register occurs. The value written is not used. Reading this register returns a zero.

27.6.7.12 CRCU Context Register

The Context Register can be written with an intermediate CRC result or desired initial state prior to processing any data. Once processing is complete, the CRC result is available from this register. Note that the CRC result is stored in the upper half of this register; the lower half is not used. The reset state of this register is all ones, because this allows the CRC32 algorithm to detect bit errors in the leading zeros of a message. A context error is generated if this register is written

after processing begins. An early read error is generated if this register is read while the EU is busy.

If the CRCU is in the default output mode (DOS and DOC bits are 0), this register holds the CRC remainder after it is bit swapped, byte swapped, and complemented. This sequence of operations is described in the protocol specifications and generates a result that can be written directly to the end of a frame or command. The result is shown in **Figure 27-7**.

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Value	x^{24}	x^{25}	x^{26}	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}	x^{16}	x^{17}	x^{18}	x^{19}	x^{20}	x^{21}	x^{22}	x^{23}	x^{8}	x^{9}	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{0}	x^{1}	x^{2}	x^{3}	x^{4}	x^{5}	x^{6}	x^{7}

Figure 27-7. CRCU Context Register Contents, Upper Half (Read—Default Mode)

If the CRCU is in the Disable Output Swap mode (DOS bit is 1), the Context Register holds the unswapped but complemented CRC remainder, as shown in **Figure 27-8**.

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Value	x^{31}	x^{30}	x^{29}	x^{28}	x^{27}	x^{26}	x^{25}	x^{24}	x^{23}	x^{22}	x^{21}	x^{20}	x^{19}	x^{18}	x^{17}	x^{16}	x^{15}	x^{14}	x^{13}	x^{12}	x^{11}	x^{10}	x^{9}	x^{8}	x^{7}	x^{6}	x^{5}	x^{4}	x^{3}	x^{2}	x^{1}	x^{0}

Figure 27-8. CRCU Context Register Contents, Upper Half (Read—DOS Mode)

If the CRCU is in the Disable Output Complement mode (DOC bit is 1), this register holds the uncomplemented but swapped CRC remainder, as shown in **Figure 27-9**.

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Value	x^{24}	x^{25}	x^{26}	x^{27}	x^{28}	x^{29}	x^{30}	x^{31}	x^{16}	x^{17}	x^{18}	x^{19}	x^{20}	x^{21}	x^{22}	x^{23}	x^{8}	x^{9}	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{0}	x^{1}	x^{2}	x^{3}	x^{4}	x^{5}	x^{6}	x^{7}

Figure 27-9. CRCU Context Register Contents, Upper Half (Read—DOC Mode)

If the CRCU is in both Disable Output Complement mode (DOC bit is 1) and Disable Output Swap mode (DOS bit is 1), this register holds the uncomplemented and unswapped CRC remainder, as shown in **Figure 27-10**. This form is the one used internally to match against the polynomial specific residue when performing ICV checking. This form can also be written back to the CRCU after reset to continue a partial CRC operation.

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Value	x^{31}	x^{30}	x^{29}	x^{28}	x^{27}	x^{26}	x^{25}	x^{24}	x^{23}	x^{22}	x^{21}	x^{20}	x^{19}	x^{18}	x^{17}	x^{16}	x^{15}	x^{14}	x^{13}	x^{12}	x^{11}	x^{10}	x^{9}	x^{8}	x^{7}	x^{6}	x^{5}	x^{4}	x^{3}	x^{2}	x^{1}	x^{0}

Figure 27-10. CRCU Context Register Contents, Upper Half (Read—DOC and DOS Mode)

27.6.7.13 CRCU Key Register

The Key Register stores the polynomial and residue for the dynamic custom mode as set in the Mode Register (**Section 27.7.13.1**). The reset value of this register is all zeros with a one in bit position 0. This register is dynamic, in that it is reset by performing a reinitialize or a software reset. This allows a custom polynomial to be used for specific processing without changing the platform-specific static custom polynomial stored in the Control Register (**Section 27.6.7.6**). A residue does not need to be programmed unless ICV checking is being performed. A context

error is generated if this register is written after processing begins. A polynomial error is generated if a value is written to this register which does not have a one in bit g^0.

27.6.7.14 CRCU FIFO

Words written to this address range are pushed onto the CRCU input FIFO, thereby buffering them for processing. Partial words and misaligned data can be written to this address and it is automatically realigned based on a big-endian byte order.

27.6.8 SNOW3G Execution Unit (STEU)

The STEU is a functional block capable of encrypting/decrypting and/or performing integrity checks on 32-bit blocks of data using a 128-bit key. The STEU is designed to accelerate the 3G UEA2 and UIA1 algorithms and the LTE EEA1 and EIA1 algorithms.

The following sections describe the STEU modes of operation, status and control registers, and FIFOs. Most of the registers are not normally accessed by the core, but are documented primarily for debug purposes. In typical operation, the STEU is used through channel-controlled access, which means that most reads and writes of STEU registers are directed by the SEC channels. Driver software performs core-controlled register accesses only on a few registers for initial configuration and error handling. Detailed programming information is provided in **Section 27.7**.

27.6.8.1 ICV Checking

In F9 mode, this EU includes an ICV checking feature, that is, it can generate an ICV and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the core either via interrupt by a writeback of EU status fields into core memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see **Section 27.7.6.1**), and mask the ICE bit in the Interrupt Mask Register (see **Section 27.7.14.7**). In the case of status writeback, the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, an error interrupt is sent to the core, but no channel done interrupt or writeback is sent.

27.6.8.2 STEU Mode Register

The Mode register determines the mode of operation. If an undefined mode is set, an Illegal Mode Error is generated. If the mode is modified during message processing, a Context Error is generated.

27.6.8.3 STEU Key Size Register

The STEU Key Size register does not physically exist, because the key size is always 16 bytes for the SNOW3G algorithms. If an illegal key size is written, an Illegal Key Size Error is generated.

27.6.8.4 STEU Data Size Register

The Data Size Register holds the number of bits to process. This value must be set prior to loading message data and after the Mode Register is written. The STEU internally decrements this value while it processes the message. The Data Size Register can be read at any time to determine the number of bits left to process. The STEU continues to process data until the value in the Data Size register reaches zero. For the F9 mode, the data size must be divisible by 64 when the PE bit is cleared in the Mode Register. In other words, the message can be split for multi-session processing only on a 64-bit boundary. If this rule is violated, an Illegal Data Size Error is generated. If 0 is written to the Data Size register, followed by EOM, or only EOM is written, the STEU asserts the done interrupt which signals that processing is complete.

27.6.8.5 STEU Reset Control Register

The Reset Control register controls the type of reset. All bits are self-clearing. Reading from this register returns 0.

27.6.8.6 STEU Status Register

The Status Register is read-only and reflects the current state of the STEU.

27.6.8.7 STEU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors occurred and generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the STEU Interrupt Mask Register is zero. If the STEU Interrupt Status Register is non-zero, the STEU halts and the STEU error interrupt signal is asserted to the controller. In addition, if the STEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in the Channel Status Register and generates a channel error interrupt to the controller. To recover and start new processing after an error, the STEU must be reset (using the SR bit in the Reset Control Register, see **Section 27.7.14.4**).

27.6.8.8 STEU Interrupt Mask Register

The Interrupt Mask Register is used to disable an error indication by writing 1 to the corresponding bit. Disabling an error causes the STEU to ignore the error and continue processing. The error is not indicated in the Interrupt Status Register and the error interrupt signal is not asserted.

Extreme care should be taken when disabling errors because it can produce invalid results. For normal operation, do not disable errors. Errors should only be disabled during debug operation.

Note that processing halts if an internal error occurs, even if the Internal Error indicator in the Interrupt Status Register is masked. An internal error is indicated when the Status Register HALT bit is set accompanied by the absence of any asserted bits in the Interrupt Status Register.

27.6.8.9 STEU Data Out Register (F9 MAC)

The Data Out Register is physically the same as the STEU Context Register 1. The difference is that the Data Out Register can be read at any time, even during processing, while the Context Register 1 can only be read after processing is completed. Writing to the Data Out Register is ignored. Reading the register returns the contents of the Context Register 1, which is used for computation of the MAC. The MAC produced by F9 mode, when read after processing is completed, always has the upper 32 bits equal to 0x0.

27.6.8.10 STEU End of Message Register

The End of Message Register indicates that all of the message data is provided to the STEU. Once the End of Message Register is written, the STEU processes any remaining data in the input FIFO and generates the done interrupt. Reading this register returns all zeros.

27.6.8.11 STEU IV_1 Register

The IV_1 register is used during the initialization phase of both F8 confidentiality and F9 integrity algorithms. The IV_1 register must be written before processing of a new message starts. Once the initialization phase is completed, the IV_1 register is no longer used in normal processing of F8 or F9 modes. Although generally this register is considered to be a context register, it does not need to be saved/restored during context switching in multi-session message processing. As with other context registers, any byte of this register can be enabled or disabled during accesses.

For f8 operations without hardware ICV checking, the IV_1 Register is the only register that must be loaded. Consequently, the Context In Length field in the STEU descriptor must be set to 8B.

For f9 operations with or without hardware ICV checking, the IV_1, ICV_In, and IV_2 registers must be loaded, and the Context In Length field in the STEU descriptor must be set to 24B. When hardware ICV checking is not enabled, the context must be constructed so that bytes 9–16 of the Context In are set to zero.

27.6.8.12 STEU ICV_In Register

If ICV checking is required, then the value to be compared with the computed f9 MAC value must be written to the ICV_In Register before Data Size is written. The 32-bit written ICV occupies the lower half of the ICV_In register, as shown in **Section 27.7.14.11**, *STEU ICV_In Register (STEUICVIR)*, on page 27-319.

Caution: *The LTE EIA1 and 3G UIA2 algorithms (based on Snow F9) produce an 8B MAC, however only the 4 MSBs of the MAC are transmitted with the PDU as the MAC-I. When operating in Snow F9 mode, the STEU is able to output the needed 4B of MAC, however its 4B MAC output is preceded by 4B of zeroes.*

Example 27-3. STEU ICV Checking

Assume that a given F9 calculation over a packet produces an 8B MAC of 0xABCDABCD_EEEEFFFF. The LTE EIA1 and 3G UIA2 algorithms transmit the packet with a MAC-I value of 0xABCDABCD. The STEU internally generates 0xABCDABCD_EEEEFFFF, but it outputs 0x00000000_ABCDABCD. If the descriptor ICV Out length is only set to 4B, only 0x00000000 is output, meaning the user is required to output all 8B of the generated ICV and only append the 4 LSBs to the packet as the MAC-I.

As a consequence, hardware ICV checking (CICV = 1) isn't practical for the STEU. The SEC channel compares Extent bytes of received MAC with the same number of bytes of MAC output by the STEU, beginning with the MSB. This leads to a miscomparison, because the received 0xABCDABCD is compared to 0x00000000. The user options are (with Extent = 0) to either copy the received 4B MAC to bytes 13–16 of the Context In data structure and set CICV = 1, or use the descriptor ICV Out field with length set to 8B, so that software can perform the comparison of the received MAC with the 4 LSBs of the STEU output (CICV = 0).

27.6.8.13 STEU IV_2 Register (FRESH)

The STEU IV_2 Register hold input context during 3G UIA2 and LTE EIA1 operations. The format of the IV_2 register for each of the algorithms is as follows.

- UIA2: 4B of zeroes || 4B Fresh (4B)
- EIA1: 4B of zeroes || 5b Bearer || 27b zeroes

The specific value is used during the initialization phase of the f9 algorithm and is ignored when the f8 algorithm is selected. This register must be written before a new message to be processed with f9 is started. Once the initialization phase is completed, the IV_2 Register is no longer used during message processing. The IV_2 Register need not be saved/restored during context switches.

27.6.8.14 STEU Context Registers

There are four 64-bit STEU Context Registers (Context 1–Context 4) that are used in F9 processing. These allow the core to interrupt F9 message processing and to resume at a later time. The Context Registers must be read when changing context and restored to their original values prior to resuming processing of an interrupted message. If the any of the STEU Context Registers are written during message processing, a Context Error is generated. All STEU Context Registers are cleared when a hard/soft reset or initialization is performed.

27.6.8.15 STEU LFSR State Registers

The STEU LFSR State Registers are used to record LFSR state during F8 processing. These registers must saved/restored when switching context in F8 mode.

27.6.8.16 STEU FSM State Registers

The STEU FSM State Registers are used to record FSM state during F8 processing. These registers must be saved/restored when switching context in F8 mode. Note that the lower half of FSM State Register 2 is not used.

27.6.8.17 STEU Key Data Registers (Confidentiality Key)

The STEU Key Data Register 1 and Key Data Register 2 together hold one 128-bit key that is used for initialization. Key Data Register 1 holds the upper 8 bytes of key data while Key Data Register 2 holds the lower 8 bytes. The STEU Key Data Registers must be written before message processing begins. Writing to either of the Key Data Registers during processing results in a Context Error. Reading from either of the Key Data Registers always returns 0.

27.6.8.18 STEU FIFOs

STEU uses an input FIFO/output FIFO pair to hold data before and after processing. Normally, the channels control all access to these FIFOs. For core-controlled operation, a write to anywhere in the STEU FIFO address space enqueues data to the input FIFO, and a read from anywhere in the STEU FIFO address space dequeues data from the output FIFO.

Writes to the input FIFO go first to a staging register that can be written by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the staging register are written, the entire 8 bytes is automatically enqueued into the FIFO. If any byte is written twice between enqueues, an Address Error results. When writing the last portion of data, it is not necessary to write all 8 bytes. Any remaining input bytes in the staging register are automatically padded with zeros and forced into the input FIFO when the STEU End of Message Register is written.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When the first 8 bytes are read, that data is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, an Address Error results.

Overflows and underflows caused by reading or writing the STEU FIFOs are reflected in the STEU Interrupt Status Register.

The STEU fetches data 64 bits at a time from the input FIFO. During F8 processing, the input data is XORed with the generated keystream and the results are placed in the output FIFO. The output size is the same as the input size. During F9 processing, the input data is hashed with the integrity key and the resulting MAC is placed in the Data Out Register.

27.6.9 Random Number Generator (RNGU)

The RNGU is an execution unit capable of generating 64-bit random numbers. It combines a True Random Number Generator (TRNG) and a deterministic Pseudo-Random Number Generator (PRNG), based on SHA, as described in FIPS 186-2, Appendix 3.1. It is designed to comply with the FIPS-140 standard for randomness and non-determinism.

The RNGU consists of five major functional blocks:

- 64-bit slave bus interface, registers, and FIFO
- True Random Number Generator (ring oscillator, Linear Feedback Shift Registers (LFSRs), and Statistical Checker)
- Xseed Generator
- Pseudo-Random Number Generator (XKEY, SHA-1, FSM)
- Simultaneous Reseed LFSR

The states of the LFSRs in the TRNG are advanced at unknown frequency determined by the ring oscillator clock. The entropy generated by this structure is then added into the XKEY structure of the PRNG during seed generation. Seed generation takes approximately 2,000,000 cycles as 20,000 bits of entropy are sampled from the output of the LFSRs of the TRNG.

After the initial seeding, the RNGU turns off the TRNG and uses solely the PRNG to generate random data. After 1,000,000 times through the algorithm the RNGU is once again seeded. This second seed occurs the next time through the algorithm by using data from the Simultaneous Reseed LFSR to modify the algorithm. The data in the simultaneous reseed LFSR comes directly from the TRNG as well and was being generated during the first 20,000 times through the PRNG algorithm after the initial seed was completed.

Most of the registers described in this section are not normally accessed by the core processor. They are documented here mainly for debug purposes. During typical operation, the RNGU is used through channel-controlled access, which means that most reads and writes of RNGU registers are directed by the SEC channels. Driver software performs core-controlled register accesses only on a few registers for initial configuration and error handling.

The following subsections include general descriptions of the RNGU registers and structures. **Section 27.7**, *Programming Model*, on page 27-95 provides a detailed description of each register and associated register fields.

27.6.9.1 RNGU Mode Register

The RNGU Mode Register is a writable location but all mode bits are currently reserved. It is documented for the sake of consistency with the other EUs.

27.6.9.2 RNGU Data Size Register

The RNGU Data Size Register is used to tell the RNGU to begin generating random data. The actual contents of the Data Size Register do not affect the operation of the RNGU. After a reset and prior to the first write of data size, the RNGU builds entropy without pushing data onto the FIFO. Once the Data Size Register is written, the RNGU begins pushing data onto the FIFO. Eight bytes (64 bits) of data are pushed onto the FIFO every 112 cycles until the FIFO is full. The RNGU then attempts to keep the FIFO full.

27.6.9.3 RNGU Reset Control Register

This register contains three reset options specific to the RNGU: Reset interrupt, module initialization, and software reset. See **Section 27.7.15.3**, *RNGU Reset Control Register (RNGRCR)*, on page 27-331 for programming details.

27.6.9.4 RNGU Status Register

This RNGU Status Register contains five fields that reflect the state of the RNGU internal signals. The RNGU Status Register is read-only. Writing to this location results in an address error being reflected in the RNGU Interrupt Status Register. See **Section 27.7.15.4**, *RNGU Status Register (RNGSR)*, on page 27-332 for details.

27.6.9.5 RNGU Interrupt Status Register

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the RNGU Interrupt Mask Register is zero (see **Section 27.7.15.5**, *RNGU Interrupt Status Register (RNGISR)*, on page 27-333).

If the RNGU Interrupt Status Register is non-zero, the RNGU halts and the RNGU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186). In addition, if the RNGU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Pointer Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the RNGU Reset Control Register.

27.6.9.6 RNGU Interrupt Mask Register

The RNGU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.7.15.5**, *RNGU Interrupt Status Register (RNGISR)*, on page 27-333), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

27.6.9.7 RNGU End_of_Message Register

The RNGU End_of_Message Register is a writable location can be used to start the RNGU to produce random numbers. A write of any value to this register will cause the RNGU to begin to produce random numbers in the FIFO. It is documented for the sake of consistency with the other EUs.

27.6.9.8 RNGU Entropy Registers

The RNGU allows the user to input Entropy into the PRNG algorithm to modify the randomness of the RNGU. This group of registers are write only and all writes to these registers are ignored when the RNGU is busy. However, when the RNGU is IDLE (FIFO is full or RNGU has not yet been started), all data written to these registers is used to modify the internal XKEY structure. These registers cannot be written back to back, there must be a clock cycle in between writes, so that the RNGU can process all 64-bits of data, as the RNGU processes only 32-bits per cycle.

27.6.9.9 RNGU FIFO

The RNGU uses an output FIFO to collect periodically sampled random 64-bit numbers, with the intent that random data is always available for reading. Normally, the channels control all access to this FIFO. For core processor-controlled operation, a read from anywhere in the RNGU FIFO address space dequeues data from the RNGU output FIFO.

The output FIFO is readable using byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, those 8 bytes are automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Underflows caused by reading or writing the RNGU output FIFO are reflected in the RNGU Interrupt Status Register. Also, a write to the RNGU output FIFO space is reflected as an addressing error in the RNGU Interrupt Status Register.

27.7 Programming Model

The SEC uses the following structures and registers to configure and operate the security engine and the execution units:

- Descriptors and Link Tables
  - Descriptor structure, **page 27-99**
  - Descriptor header, **page 27-100**
  - Descriptor Pointers, **page 27-173**
  - Link tables, **page 27-175**
- SEC Controller
  - Master Control Register (MCR), **page 27-177**
  - Controller Identification Register (CIDR), **page 27-180**
  - Controller IP Block Revision Register (CIPBRR), **page 27-180**
  - EU Assignment Status Register (EUASR), **page 27-181**
  - Controller Interrupt Enable Register (CIER), **page 27-182**
  - Controller Interrupt Status Register (CISR), **page 27-186**
  - Controller Interrupt Clear Register (CICR), **page 27-189**
- Polychannel
  - Fetch FIFO Enqueue Counter (FFEC), **page 27-192**
  - Descriptor Finished Counter (DFC), **page 27-193**
  - Data Bytes In Counter (DBIC), **page 27-194**
  - Data Bytes Out Counter (DBOC), **page 27-195**
- Channels
  - Channel 1–4 Configuration Registers (CCR[1–4], **page 27-196**
  - Channel 1–4 Pointer Registers (CSR[1–4]), **page 27-199**
  - Channel 1–4 Current Descriptor Pointer Registers (CDPR[1–4]), **page 27-204**
  - Channel 1–4 Fetch FIFOs (CFF[1–4], **page 27-205**
  - Channel 1–4 Descriptor Buffers, **page 27-206**
- PKEU
  - PKEU Mode Register (PKEUMR), **page 27-207**
  - PKEU Key Size Register (PKEUKSR), **page 27-209**
  - PKEU AB Size Register (PKEUABSR), **page 27-210**
  - PKEU Data Size Register (PKEUDSR), **page 27-211**
  - PKEU Reset Control Register (PKEURCR), **page 27-212**
  - PKEU Status Register (PKEUSR), **page 27-213**
  - PKEU Interrupt Status Register (PKEUISR), **page 27-214**
  - PKEU Interrupt Mask Register (PKEUIMR), **page 27-216**
  - PKEU End_of_Message Register (PKEUEOMR), **page 27-217**
  - PKEU Parameter Memories, **page 27-217**

- DEU
  - DEU Mode Register (DEUMR), **page 27-219**
  - DEU Key Size Register (DEUKSR), **page 27-220**
  - DEU Data Size Register (DEUDSR), **page 27-221**
  - DEU Reset Control Register (DEURCR), **page 27-222**
  - DEU Status Register (DEUSR), **page 27-223**
  - DEU Interrupt Status Register (DEUISR), **page 27-224**
  - DEU Interrupt Mask Register (DEUIMR), **page 27-226**
  - DEU End_of_Message Register (DEUEOMR), **page 27-228**
  - DEU IV Register (DEUIVR), **page 27-228**
  - DEU Key Registers (DEUKR[1–3], **page 27-229**
  - DEU Input FIFO / Output FIFO, **page 27-229**
- AESU
  - AESU Mode Register (AESUMR), **page 27-230**
  - AESU Key Size Register (AESUKSR), **page 27-233**
  - AESU Data Size Register (AESUDSR), **page 27-234**
  - AESU Reset Control Register (AESURCR), **page 27-235**
  - AESU Status Register (AESUSR), **page 27-236**
  - AESU Interrupt Status Register (AESUISR), **page 27-237**
  - AESU Interrupt Mask Register (AESUIMR), **page 27-239**
  - AESU ICV Size Register (AESUICVSR), **page 27-241**
  - AESU End_of_Message Register (AESUEOMR), **page 27-242**
  - AESU Context Registers (AESUCR[1–12]), **page 27-243**
  - AESU Key Registers (AESUKR[1–2]), **page 27-246**
  - AESU Input FIFO/Output FIFO, **page 27-247**
- MDEU
  - MDEU Mode Register (MDEUMR), **page 27-248**
  - MDEU Key Size Register (MDEUKSR), **page 27-250**
  - MDEU Data Size Register (MDEUDSR), **page 27-251**
  - MDEU Reset Control Register (MDEURCR), **page 27-252**
  - MDEU Status Register (MDEUSR), **page 27-253**
  - MDEU Interrupt Status Register (MDEUISR), **page 27-254**
  - MDEU Interrupt Mask Register (MDEUIMR), **page 27-256**
  - MDEU ICV Size Register (MDEUICVSR), **page 27-258**
  - MDEU End_of_Message Register (MDEUEOMR), **page 27-259**
  - MDEU Context Registers (MDEUCR), **page 27-260**
  - MDEU Key Registers (MDEUKR[1–8]), **page 27-262**
  - MDEU Input FIFO, **page 27-262**
- AFEU
  - AFEU Mode Register (AFEUMR), **page 27-263**

- CRCU Input FIFO, **page 27-306**

- STEU
  - STEU Mode Register (STEUMR), **page 27-307**
  - STEU Key Size Register (STEUKSR), **page 27-308**
  - STEU Data Size Register (STEUDSR), **page 27-309**
  - STEU Reset Control Register (STEURCR), **page 27-310**
  - STEU Status Register (STEUSR), **page 27-311**
  - STEU Interrupt Status Register (STEUISR), **page 27-312**
  - STEU Interrupt Mask Register (STEUIMR), **page 27-314**
  - STEU Data Out Register (STEUDOR), **page 27-316**
  - STEU End_of_Message Register (STEUEOMR), **page 27-317**
  - STEU IV1 Register (STEUIV1R), **page 27-318**
  - STEU ICV_In Register (STEUICVIR), **page 27-319**
  - STEU IV2 Register (STEUIV2R), **page 27-320**
  - STEU Context Registers 1 (STEUCR1), **page 27-321**
  - STEU Context Registers 2 (STEUCR2), **page 27-322**
  - STEU Context Registers 3 (STEUCR3), **page 27-323**
  - STEU Context Registers 4 (STEUCR4), **page 27-324**
  - STEU LFSR State Register 0–7 (STEULFSRSR[0–7]), **page 27-325**
  - STEU FSM State Register 1 (STEUFSMSR1), **page 27-326**
  - STEU FSM State Register 2 (STEUFSMSR2), **page 27-327**
  - STEU Key Data Registers 1–2 (STEUKDR[1–2]), **page 27-328**
  - STEU Input FIFO/Output FIFO, **page 27-328**
- Random Number Generator (RNGU)
  - RNGU Mode Register (RNGMR), **page 27-329**
  - RNGU Data Size Register (RNGDSR), **page 27-330**
  - RNGU Reset Control Register (RNGRCR), **page 27-331**
  - RNGU Status Register (RNGSR), **page 27-332**
  - RNGU Interrupt Status Register (RNGISR), **page 27-333**
  - RNGU Interrupt Mask Register (RNGIMR), **page 27-334**
  - RNGU End_of_Message Register (RNGEOMR), **page 27-336**
  - RNGU Entropy Registers 0–7 (RNGER[0–7], **page 27-336**
  - RNGU Output FIFO, **page 27-337**

Note: Offsets used with the memory structures and registers in the SEC are from the CCSR base address 0xFFF10000.

27.7.1 Descriptors and Link Tables

The following sections described the overall descriptor structure, the descriptor header, descriptor pointers, and the link table format.

27.7.1.1 Descriptor Structure

Bit	63 48	47	46 40	39 36	35 32	31 0
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Length0	J0	Extent0	—	Eptr0	Pointer0
Pointer 1	Length1	J1	Extent1	—	Eptr1	Pointer1
Pointer 2	Length2	J2	Extent2	—	Eptr2	Pointer2
Pointer 3	Length3	J3	Extent3	—	Eptr3	Pointer3
Pointer 4	Length4	J4	Extent4	—	Eptr4	Pointer4
Pointer 5	Length5	J5	Extent5	—	Eptr5	Pointer5
Pointer 6	Length6	J6	Extent6	—	Eptr6	Pointer6

SEC descriptors are designed to support the cryptographic computation of a single packet using a single descriptor. They are conceptually similar to descriptors used by most devices with DMA capability. The descriptors have a fixed length of 64 bytes. A descriptor consists of one header (8 bytes) and seven pointers (each 8 bytes).

The upper half of the header is a Descriptor Control field, and the lower half is the Descriptor Feedback field. The Descriptor Control field of the header specifies the security operation to be performed, the execution unit(s) needed, and the modes for each execution unit. The pointers, all of which have the same format, contain pointer and length information for locating input or output parcels (such as keys, context, or text-data). The large number of pointers provided in the descriptor allows for multi-algorithm operations that require fetching of multiple keys, as well as fetch and return of contexts. Any pointer that is not needed can be given a length of zero.

SEC descriptors include scatter/gather capability, which means that each pointer in a descriptor can be either a direct pointer to a contiguous parcel of data, or can be a pointer to a link table which is a list of pointers and lengths used to assemble the parcel. When a link table is used to read input data, this is referred to as a gather operation; when used to write output data, it is referred to as a scatter operation.

27.7.1.2 Descriptor Header

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	OP_0												OP_1												DESC_TYPE					—	DIR	DN
	EU_SEL0				MODE0								EU_SEL1				MODE1															
Writeback	DONE												—																			

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
Field	—																															
Writeback	—			ICCR0				—				ICCR1				—									ID_TAG							

The header specifies the security operation to perform, the execution unit(s) needed, and the modes for each execution unit. **Table 27-13** describes the fields that must be supplied to the SEC. The SEC device drivers allow the core processor to create proper headers for each cryptographic operation. SEC descriptor processing sometimes includes performing a writeback, that is, writing the original header back to system memory with certain fields modified. The modified fields are shown in the Writeback rows and described in **Table 27-14**.

Table 27-13. Header Bit Definitions

Bits	Name		Description	Settings
63–60	OP_0	EU_SEL0	**Primary EU Select** **Notes:** **1.** EU_SEL0 values of “No EU selected” or “Reserved” results in an “Unrecognized Header Error” condition during processing of the descriptor header. **2.** If EU_SEL1 is CRCU or MDEU, then EU_SEL0 must be DEU, AESU, AFEU, KEU, or STEU. All other values of EU_SEL0 result in an “Unrecognized header” error condition.	See **Table 27-15** for setting values.
59–52		MODE0	**Primary Mode** Mode data used to program the primary EU. The mode data is to the chosen EU. This field is passed directly to bits 7–0 of the Mode Register in the selected EU.	
51–48	OP_1	EU_SEL1	**Secondary EU Select** **Notes:** **1.** The only valid choices for EU_SEL1 are “No EU selected”, CRCU, or MDEU. Any other choice results in an “Unrecognized Header” error condition. **2.** If EU_SEL1 is CRCU or MDEU, then EU_SEL0 must be DEU, AESU, AFEU, KEU, or STEU. All other values of EU_SEL0 result in an “Unrecognized header” error condition.	See **Table 27-15** for setting values.
47–40		MODE1	**Secondary Mode** Mode data used to program the primary EU. The mode data is to the chosen EU. This field is passed directly to bits 7–0 of the Mode Register in the selected EU.	

Table 27-13. Header Bit Definitions (Continued)

Bits	Name	Description	Settings
39–35	DESC_TYPE	**Descriptor Type** Along with the DIR field, this determines the sequence of actions to be performed by the channel and selected EUs using the blocks of data listed in the rest of the descriptor. The determined attributes include: • the direction of data flow for each data block • which EU (primary or secondary) to access • what snooping options to use • and which internal EU addresses to access	See **Table 27-16** for valid settings. **Table 27-17** shows how the pointers should be used with the various descriptor types to load keys, context, and text-data into the Execution Units, and how the required outputs should be unloaded.
34	—	Reserved.	
33	DIR	**Direction: Direction of Overall Data Flow** Along with the DESC_TYPE field, this field helps determine the sequence of actions to be performed by the channel and selected EUs.	0 Outbound 1 Inbound
32	DN	**Done Notification** This enables done notification if the NT field is 1 in the Channel Configuration Register (see **Table 27-15**). The done notification can take the form of an interrupt, a writeback in the DONE field of this header (see **Table 27-16**), or both, depending upon the states of the Channel Done Interrupt Enable (CDIE) and Channel Done Writeback Enable (CDWE) bits in the Channel Configuration Register.	0 No done notification. 1 Signal DONE to the core processor on completion of this descriptor.
31–0	—	Reserved	

Table 27-14. Header Writeback Field Definitions

Bits	Name	Description	Settings
63–56	DONE	**DONE** When DONE writeback is used, then at the completion of descriptor processing this byte is written with the value 0xFF. To determine when done writeback is used, see the CDWE, NT, and CDIE fields in the Channel Configuration Register (**Section 27.7.6.1**, *Channel Configuration Registers for Channels 1–4 (CCR[1–4]),* on page 27-196).	
55–29	—	Reserved.	
28–27	ICCR0	**Integrity Check Comparison Result from Primary** These bits are supplied by the primary EU when descriptor processing is complete.	00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
26–21	—	Reserved.	
20–19	ICCR1	**Integrity Check Comparison Result from Secondary** These bits are supplied by the secondary EU (if any) when descriptor processing is complete.	00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved.

Table 27-14. Header Writeback Field Definitions (Continued)

Bits	Name	Description	Settings
18–16	—	Reserved.	
15–0	ID_TAG	Identification Tag This value is copied from the ID_TAG field written by the host into the Fetch FIFO (see **Section 27.7.6.4**, *Channel Fetch FIFO (CFF)*).	

Table 27-15. EU_SEL0 and EU_SEL1 Values

Value (binary)	Selected EU
0000	No EU selected
0001	AFEU (not valid for EU_SEL1)
0010	DEU (not valid for EU_SEL1)
0011	MDEU-A
0100	RNGU (not valid for EU_SEL1)
0101	PKEU (not valid for EU_SEL1)
0110	AESU (not valid for EU_SEL1)
0111	KEU (not valid for EU_SEL1)
1000	CRCU
1001	STEU (not valid for EU_SEL1)
1010	Reserved
1011	MDEU-B
1100–1110	Reserved
1111	Reserved for header writeback

Note: MDEU has two different designators to refer to the same Message Digest Execution Unit. If MDEU-B is selected, then the Channel configures MDEU to perform SHA-224, SHA-256, SHA-384, and SHA-512. If MDEU-A is selected, then the Channel configures MDEU to perform SHA-160, SHA-224, SHA-256, or MD5. The channel uses the MODE0 (or MODE1) value to configure the MDEU Mode Register (MDEUMR) if it is selected by the descriptor and this automatically sets or clears the MDEUMR[MDEU_B] bit to select MDEU-A or MDEU-B.

27.7.1.2.1 Descriptor Types

Table 27-16 lists the available descriptors used by the SEC to identify the encryption operations to perform.

Table 27-16. Descriptor Types

Value (binary)	Descriptor Type	Notes
Descriptor Types (least significant bit must be 0)		
00000	aesu_ctr_nonsnoop	AESU-CTR non snooping operations.
00010	common_nonsnoop	Common, non snooping, non-PKEU, non-AFEU. For use with AES-CTR, you must prepend zeros to the AES-Context before loading the AES Context Registers.
00100	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU. For use with AES-CTR with HMAC, you must prepend zeros to the AES-Context before loading the AES Context Registers.
00110	—	Reserved
01000	—	Reserved
01010	common_nonsnoop_afeu	Common, non snooping, AFEU

Table 27-16. Descriptor Types (Continued)

Value (binary)	Descriptor Type	Notes
01100	—	Reserved
01110	—	Reserved
10000	pkeu_mm	PKEU-Montgomery Multiplication
10010	—	Reserved
10100	—	Reserved
10110	—	Reserved
11000	hmac_snoop_aesu_ctr	AESU CTR with HMAC snooping
11010	—	Reserved
11100	—	Reserved
11110	—	Reserved
Descriptor Types (least significant bit must be 1)		
00001	ipsec_esp	IPsec ESP mode encryption and hashing
00011	802.11i AES ccmp	CCMP encryption and hashing, suitable for 802.11i
00101	srtp	SRTP encryption and hashing
00111	pkeu_build	pkeu_build Elliptic Curve Cryptography
01011	pkeu_ptadd_dbl	pkeu_ptadd_dbl Elliptic Curve Cryptography
01101	—	Reserved
01111	—	Reserved
10001	tls_ssl_block	TLS/SSL generic block cipher
10011	tls_ssl_stream	TLS/SSL generic stream cipher
10101	raid_xor	XOR 3 sources together
10111	ipsec_aes_gcm	IPsec ESP mode using AES GCM encryption and hashing
11001	dbl_crc	Do two Cyclic Redundancy Check Operations
11011	—	Reserved
11101	—	Reserved
11111	—	Reserved

27.7.1.2.2 Descriptor Formats

Table 27-17 summarizes how the descriptors can be used.

Table 27-17. Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer 1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0000_0 aesu_ctr_ nonsnoop	Length	reserved	Cipher Context In	Cipher Key	Main Data In	Data Out	Cipher Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nonsnoop for DES, KEU f8, STEU f8, STEU f9, RNGU, AES-CCM	Length	reserved	Context In	Key	Main Data In	Data Out	Context Out (incl. ICV Out)	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nonsnoop for MDEU	Length	reserved	Context In	Key	Main Data In	ICV In	Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nonsnoop for STEU f9, AES-XCBC, AES-CMAC	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	ICV In	reserved	reserved
0001_0 common_ nonsnoop for KEU f9,	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nonsnoop for CRCU	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	ICV In	reserved	reserved
0010_0 hmac_snoop _no_afeu	Length	Hash Key	Hash-only Header	Cipher Key	Cipher Context In	Main Data In	Data Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0101_0 common_ nonsnoop_ afeu	Length	reserved	Context In (via In FIFO)	Cipher Key	Main Data In	Data Out	Context Out (via Out FIFO)	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1000_0 pkeu_mm	Length	“N” In	“B” In	“A” In	“E” In	“B” Out	reserved	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1100_0 hmac_snoop_ aesu_ctr	Length	Hash Key	Hash-only Header	AES Key	AES Context In	Main Data In	Data Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Table 27-17. Descriptor Format Summary (Continued)

Descriptor Type	field type	Pointer0	Pointer 1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0000_1 ipsec_esp	Length	HMAC Key	Hash-only Header	Cipher IV In	Cipher Key	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	reserved	ICV In	ICV Out	reserved
0001_1 802.11i AES ccmp	Length	CRC-only Header	AES Context In	AES Key	Hash-only Header	Main Data In	Data Out	AES Context Out
	Extent	CRC In/Out (FCS)	reserved	reserved	reserved	MIC In	MIC Out	reserved
0010_1 srtp with ICV Check	Length	HMAC Key	AES Context In	AES Key	Main Data In	Data Out	HMAC Out	AES Context Out
	Extent	reserved	reserved	reserved	Hash-only Header	Hash-only Trailer	reserved	reserved
0010_1 srtp without ICV Check	Length	HMAC Key	AES Context In	AES Key	Main Data In	HMAC In	Data Out	AES Context Out
	Extent	reserved	reserved	reserved	Hash-only Header	Hash-only Trailer	HMAC Out	reserved
0011_1 pkeu_build	Length	“A0” In	“A1” In	“A2” In	“A3” In	“B0” In	“B1” In	"Build" Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0100_1 pkeu_ptmul	Length	“N” In	“E” In	“Build” In	“B1” Out	“B2” Out	“B3” Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0101_1 pkeu_ptadd_dbl	Length	“N” In	“Build” In	“B2” In	“B3” In	“B1” Out	“B2” Out	“B3” Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1000_1 outbound tls_ssl_ block	Length	MAC Key	Cipher IV In	Cipher Key	Main Data In	reserved	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV Out	reserved	reserved
1000_1 inbound tls_ssl_ block	Length	MAC Key	Cipher IV In	Cipher Key	reserved	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV In	ICV Out	reserved
1001_1 outbound tls_ssl_ stream	Length	MAC Key	Cipher IV In	Cipher Key	Main Data In	reserved	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV Out	reserved	reserved
1001_1 inbound tls_ssl_ stream	Length	MAC Key	Cipher IV In	Cipher Key	reserved	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV In	ICV Out	reserved
1010_1 raid_xor	Length	Source F Data In	Source E Data In	Source D Data In	Source C Data In	Source B Data In	Source A Data In	Data Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Table 27-17. Descriptor Format Summary (Continued)

Descriptor Type	field type	Pointer0	Pointer 1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1011_1 ipsec_aes_gcm	Length	AES Context In	AAD In	Nonce Part 2 In	AES Key In	Main Data In	Data Out	Cipher Context Out
	Extent	reserved	reserved	reserved	Nonce Part 1 In	AES ICV In	AES ICV Out	CRC ICV In/Out
1100_1 dbl_crc	Length	Header In	Payload In	reserved	reserved	reserved	reserved	reserved
	Extent	Header ICV	Payload ICV	Header ICV Out	Payload ICV Out	reserved	reserved	reserved
others		Reserved						

27.7.1.2.3 Descriptor Operations During Cryptographic Processing

The user should read and understand the following guidelines for correct use of the SEC in a particular application:

- The steps and diagrams in this document do not cover all the actions needed to implement a particular protocol, only those relevant to operations performed by the SEC. The INPUT and OUTPUT packet formats shown in the diagrams are typically not the complete packet formats handled by the protocol. Refer to the specific protocol for detailed information.
- In all cases that use two EUs, the cipher EU is set as the primary EU. The secondary EU may be MDEU for authentication, or CRCU for generating CRCs.
- The pointers obtained from descriptors are stored in the channel and are incremented as reads or writes take place. If the steps specify that a channel uses a pointer a second time (to access a second parcel), these accesses continue where the first operation left off. For example, the steps might call for reading E3 bytes starting at pointer P3 to obtain hash-only data. This leaves P3 at the *end* of the hash-only data. In a subsequent step, pointer P3 might be used again to read the main payload, with length given by L3, and then again to read a trailer, with length given by E4.
- Setting the J-bit enables the scatter/gather feature for all parcels accessed through the pointer in the same entry. If J<i> is false, then P<i> points directly to the parcel; if J<i> is true, P<i> is a pointer to a link table. If P<i> is used to access multiple parcels, then all of those accesses are affected by J<i>.
- If a length field is zero, the length field is not used. If an extent field is zero, then any reading or writing step that uses this extent field is skipped.
- Unless otherwise noted, any padding required by the authentication function is provided by the authentication EU.
- Where automatic ICV comparison is used, there is no need to output the new hash result. Software typically specifies a length of 0 to suppress the ICV output.

- Realignments performed by SEC are not detailed here. The general rule is that all data fed to any EUs input FIFO is realigned so that the FIFO obtains a sequence of completely filled words. In some cases the last word may be only partially filled.
- The terminology in the diagrams generally reflects that used in the protocol standards. Note that different protocols use different terms for similar quantities. For example:
  - Hash, MAC, HMAC, and ICV all refer to values used for authentication.
  - IV and context both refer to values that need to be loaded into a EU before it can process data.

Figure 27-11is a key to the packet pointer diagrams used to explain the individual descriptor types in the subsequent subsections. The diagram shows a flow of activity from top to bottom.

Input parcels, denoted by pointers (P), lengths (L), and extents (E), are shown at the top of the diagram, and output parcels are shown similarly at the bottom. These have the lightest shading (green). Where a length is given in the descriptor but there is no corresponding memory data, the shading is omitted (see E3 to the right). The green shaded areas depict the layout of data in memory if link tables are not in use.

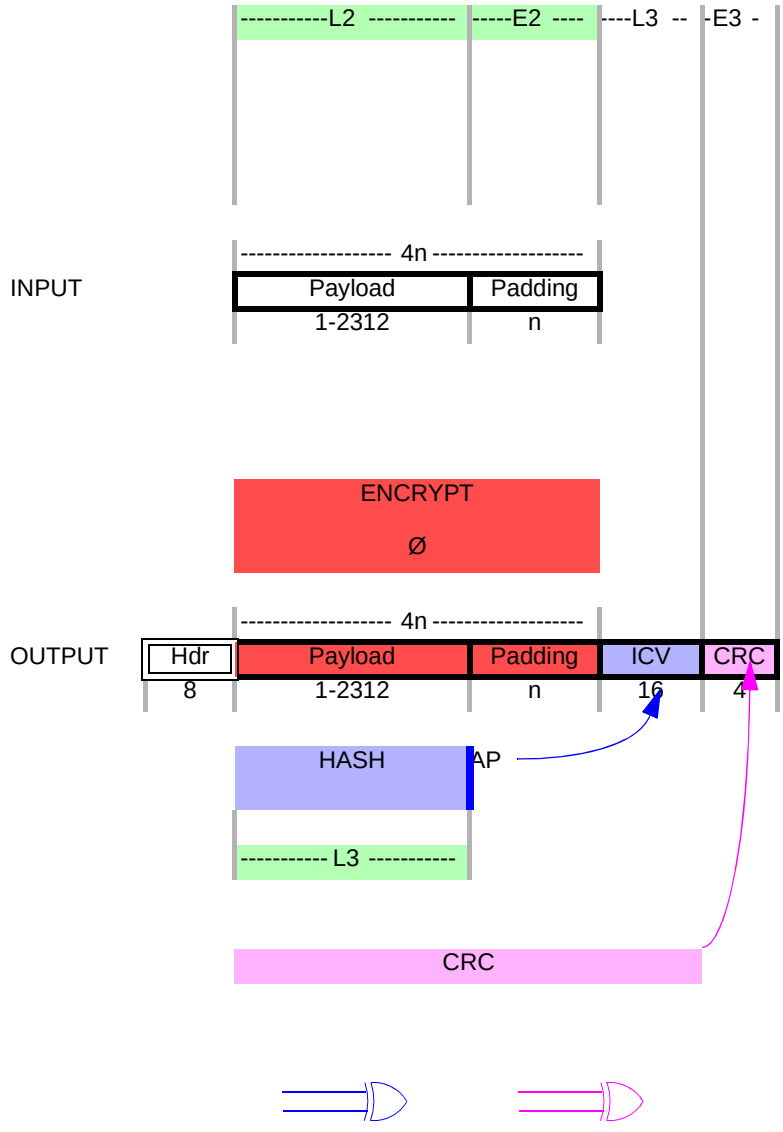
Relevant input data fields are shown with lengths labeled below them. Any additional restrictions on field sizes are shown above the fields. The following size designators are used:
B - bytes (the default for all lengths)
b - bits
n - any positive integer
4n - integer multiple of 4

An encryption or decryption process is denoted by a borderless rectangle of the darkest shading (red). Encrypted fields also have this shading.

A hashing process for authentication is denoted by a borderless rectangle of medium shading (blue). Authentication result fields also have this shading. A blue “AP” at the right end of a “HASH” rectangle means the EU provides autopadding to complete the last data block. A blue arrow shows where the authentication result is placed.

CRC operations are shown in the same way as hashing, but with magenta coloring.

A comparison between an integrity check field in a packet and a hash or CRC result is indicated by an XOR gate.



To provide additional information about the protocols, fields that are not encrypted or hashed are sometimes included in the diagrams. They are shown in white with a double border (see the “Hdr” field above).

Figure 27-11. Key to Packet Pointer Diagrams

27.7.1.2.4 Descriptor Types 0000_0: aesu_ctr_nonsnoop

Table 27-18. aesu_ctr Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0000_0 aesu_ctr_ nonsnoop	Length	reserved	Cipher Context In	Cipher Key	Main Data In	Data Out	Cipher Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Descriptor type aesu_ctr_nonsnoop is designed for use with AESU for Counter mode operation. For this descriptor type, the Channel accesses AESU Context Registers differently than most other descriptor types, permitting the transfer of a smaller AESU context when required. Descriptor type hmac_snoop_aesu_ctr (see Section 27.7.1.2.9) provides similar context access while snooping ciphertext to a secondary EU (either CRCU or MDEU).

Data Processing Steps for descriptor type aesu_ctr (as managed by the Channel):

1. AESU Mode, Key Size, Key, and Data Size registers are configured as appropriate.
2. Counter: driver builds Context in structure with initial counter value and counter modulus. Channel fetches this from address *P1* and writes into AESU Context registers.
3. Encrypt / Decrypt Input: *L3* bytes fetched from address *P3* and written to AESU input FIFO.
4. Output: *L4* bytes fetched from AESU output FIFO are written to address *P4*.
5. Counter out: *L5* bytes read from AESU context registers are written to address *P5*. This is useful should a subsequent descriptor need to start where this descriptor left off.

Table 27-19. Descriptor Control word examples for aesu_ctr

Control Word (32 bit hex)	Description
60600000	aesu_ctr nonsnoop outbound
606000002	aesu_ctr nonsnoop inbound

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	--	J0	--	-	Eptr0	P0: --
Pointer 1	AESU Context Len	J1	--	-	Eptr1	P1: Address of AESU Context In
Pointer 2	AESU Key Length	J2	--	-	Eptr2	P2: Address of AESU Key
Pointer 3	Main Data Length	J3	--	-	Eptr3	P3: Address of Main Data
Pointer 4	Main Data Out Len	J4	--	-	Eptr4	P4: Address of Main Data Out
Pointer 5	AESU Context Len	J5	--	-	Eptr5	P5: Address of AESU Context Out
Pointer 6	--	J6	--	-	Eptr6	P6: --
	Length	J	Extent	--	Eptr	Pointer

Figure 27-12. aesu_ctr_nonsnoop Descriptor Format

27.7.1.2.5 Descriptor Type 0001_0: common_nonsnoop

common_nonsnoop is the descriptor type for performing most single-EU operations, to provide raw security functionality when a protocol-specific descriptor type is either undesirable or unavailable. For Public Key Operations, there are specific descriptor types (see Section 27.7.1.2.8 for regular modular math-type operations or Section 27.7.1.2.20 for elliptic curve-type operations). In addition, a specific descriptor type has been specified for AFEU (see Section 27.7.1.2.7).

Table 27-20. common_nonsnoop Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0001_0 common_ nonsnoop for MDEU	Length	reserved	Context In	Key	Main Data In	ICV In	Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nonsnoop for KEU f9, STEU f9, AES-XCBC , AES-CMAC	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	ICV In	reserved	reserved
0001_0 common_ nonsnoop for CRCU	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	ICV In	reserved	reserved
0001_0 common_ nonsnoop for other cases not listed above	Length	reserved	Context In	Key	Main Data In	Data Out	Context Out (incl. ICV Out)	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Data Processing Steps (as managed by the Channel):

1. Setup: Mode, Key, Key Size and Data Size registers are configured as appropriate.
2. Data In: *L3* bytes are fetched from address *P3* and written into the selected EU input FIFO.
3. Data Out: If a symmetric cipher has been selected (or for RNG output data), read *L4* bytes from output FIFO and write to address *P4*.
4. Context Out: *L5* bytes read from Context registers are written to address *P5*. The definition of context depends on the execution unit; please see the section describing the Context registers for the selected execution unit for more information.
5. Integrity Check Value: In some execution units, the Integrity Check Value is part of context out. In others, it is separate.

— For AESU, KEU and STEU, *L6* bytes of ICV are read from the execution unit and written to address *P6*.
— For CRCU and MDEU, *L5* bytes of ICV are read from the execution unit and written to address *P5*.
  • For CRCU, the Context *is* the ICV.
  • For MDEU, the Context includes an additional 8-byte field. The ICV is the first N bytes in the Context, where N is dependent on which hash algorithm has been selected.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	--	J0	--	-	Eptr0	P0: unused
Pointer 1	Context In Length	J1	--	-	Eptr1	P1: Address of Execution Unit Context In
Pointer 2	Key Length	J2	--	-	Eptr2	P2: Address of Key
Pointer 3	Main Data Length	J3	--	-	Eptr3	P3: Address of Main Data In
Pointer 4	Main Data Length	J4	--	-	Eptr4	P4: Address of Main Data Out
Pointer 5	Context Out Length	J5	--	-	Eptr5	P5: Address of Execution Unit Context Out
Pointer 6	Computed ICV Len[1]	J6	--	-	Eptr6	P6: Address of Computed ICV Out
	Length	J	Extent	--	Eptr	Pointer

Figure 27-13. common_nonsnoop_no_afeu Descriptor Format

1. ICV Length and ICV Output address are used only when the execution unit provides the ICV result in a register other than a context register -- those are AESU and KFEU

Table 27-21 lists some selected functions that can be performed using the common-nonsnoop-no-afeu descriptor type, along with appropriate example Descriptor header control word values. It should be noted that this list is by no means exhaustive, and more control words can be constructed by varying the EU_MODE_0 field. For example, for descriptor triple-des-ecb encrypt, the control word is 0x20300010. In that control word, the field occupied by the value “2” identifies the Execution Unit desired, in this case DESU. The field occupied by the value “03” identifies the Mode that the channel will write into the execution unit as part of the execution unit configuration phase.

Table 27-21. Descriptor Control Word Examples For Common-nonsnoop-no-afeu Descriptors

Control Word (32 bit hex)	Description
20000012	des-ecb decrypt
20100010	des-ecb encrypt
20200012	triple-des-ecb decrypt

Table 27-21. Descriptor Control Word Examples For Common-nonsnoop-no-afeu Descriptors (Continued)

Control Word (32 bit hex)	Description
20300010	triple-des-ecb encrypt
20400012	des-cbc decrypt
20500010	des-cbc encrypt
20600012	triple-des-cbc decrypt
20700010	triple-des-cbc encrypt
20800012	des-cfb decrypt
20900010	des-cfb encrypt
20a00012	triple-des-cfb decrypt
20b00010	triple-des-cfb encrypt
20c00012	des-ofb decrypt
20d00010	des-ofb encrypt
20e00012	triple-des-ofb decrypt
20f00010	triple-des-ofb encrypt
31400010	sha-1 hash
31500010	sha-256 hash
31600010	md5 hash
31700010	sha-224 hash
31c00010	hmac-sha1
31d00010	hmac-sha-256
31e00010	hmac-md5
31f00010	hmac-sha-224
33400010	ssl-sha-1 mac
33600010	ssl-md5 mac
34c00012	hmac-sha-1 with ICV check
34e00012	hmac-md5 with ICV check
34f00012	hmac-sha-224 with ICV check
35400012	sha-1 with ICV check
35500012	sha-256 with ICV check
35600012	md5 with ICV check
35c00012	hmac-sha-1 with ICV check
35d00012	hmac-sha-256 with ICV check
35e00012	hmac-md5 with ICV check
37400012	ssl-sha-1 with ICV check
37600012	ssl-md5 with ICV check

Table 27-21. Descriptor Control Word Examples For Common-nonsnoop-no-afeu Descriptors (Continued)

Control Word (32 bit hex)	Description
38000012	hmac-sha-1 continuation
38200012	hmac-md5 continuation
39800012	hmac-sha-1 continuation
39a00012	hmac-md5 continuation
39b00010	hmac-sha-224 continuation
40000010	random number generation
60000012	aes-ecb decrypt
60100010	aes-ecb encrypt
60200012	aes-cbc decrypt
60300010	aes-cbc encrypt
60400012	aes-ofb decrypt
60500010	aes-ofb encrypt
60600000	aes-ctr cipher
62200012	aes-cbc-rbp decrypt
62300010	aes-cbc-rbp encrypt
64200012	aes-xts decrypt
64300010	aes-xts encrypt
64400010	aes-cmac integrity tag generation
68400010	aes-xcbc-mac integrity tag generation
68600012	aes-cfb decrypt
68700010	aes-cfb encrypt
6b000012	aes-cfb decrypt
6b100010	aes-ccm encrypt
6f000012	aes-ccm inbound
71800012	kasumi f8 decrypt
71800010	kasumi f8 encrypt
71a00010	kasumi f9 integrity tag generation
73800010	kasumi-f8-edge encrypt
73800012	kasumi-f8-edge decrypt
75a00012	kasumi f9 with ICV check
79800012	kasumi f8-gsm decrypt
79800010	kasumi-f8-gsm encrypt
80000010	IEEE 802 (ethernet) 32-bit crc
80100010	IETF-3385 (iSCSI) 32-bit crc

Table 27-21. Descriptor Control Word Examples For Common-nonsnoop-no-afeu Descriptors (Continued)

Control Word (32 bit hex)	Description
84000012	IEEE 802 (ethernet) 32-bit crc with ICV check
84100012	IETF-3385 (iSCSI) 32-bit crc with ICV check
90900012	snow f8 decrypt
90900010	snow f8 encrypt
91a00010	snow f9 integrity tag generation
95a00012	snow f9 with ICV check
b0800010	hmac-sha-384
b0a00010	hmac-sha-512
b0c00010	hmac-sha-384
b0e00010	hmac-sha-512
b1000010	sha-384 hash
b1200010	sha-512 hash
b1400010	sha-384 hash
b1500010	sha-256 hash
b1600010	sha-512 hash
b1700010	sha-224 hash
b1800010	hmac-sha-384
b1a00010	hmac-sha-512
b1c00010	hmac-sha-384
b1d00010	hmac-sha-256
b1e00010	hmac-sha-512
b1f00010	hmac-sha-224
b4800012	hmac-sha-384 with ICV check
b4a00012	hmac-sha-512 with ICV check
b4c00012	hmac-sha-384 with ICV check
b4d00012	hmac-sha-256 with ICV check
b4e00012	hmac-sha-512 with ICV check
b5000012	sha-384 with ICV check
b5200012	sha-512 with ICV check
b5400012	sha-384 with ICV check
b5600012	sha-512 with ICV check
b5800012	hmac-sha-384 with ICV check
b5a00012	hmac-sha-512 with ICV check
b5c00012	hmac-sha-384 with ICV check

Table 27-21. Descriptor Control Word Examples For Common-nonsnoop-no-afeu Descriptors (Continued)

Control Word (32 bit hex)	Description
b5e00012	hmac-sha-512 with ICV check
b8000010	hmac-sha-384 continuation
b8100010	hmac-sha-256 continuation
b8200010	hmac-sha-512 continuation
b8300010	hmac-sha-224 continuation
b9800010	hmac-sha-384 continuation
b9900010	hmac-sha-256 continuation
b9a00010	hmac-sha-512 continuation
b9b00010	hmac-sha-224 continuation

27.7.1.2.6 Descriptor Type 0010_0: hmac_snoop_no_afeu

Descriptor type hmac_snoop_no_afeu is designed for use when multi-processing is needed and a protocol-specific descriptor is either unavailable or undesirable. Despite the prefix “hmac”, the descriptor type can be used when either CRCU or MDEU is the secondary execution unit, and MDEU operation is not restricted to HMAC. The construction of this descriptor permits separate confidentiality and integrity keys, and also allows for a portion of data that is hashed (CRC’d) without being ciphered.

Table 27-22. Descriptor Format Summary for hmac_snoop_no_afeu

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0010_0 hmac_snoop_no_afeu	Length	Hash Key	Hash-only Header	Cipher Key	Cipher Context In	Main Data In	Data Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Data Processing Steps (as managed by the Channel):

1. Mode, Key Size, Key, and Data Size registers are configured as appropriate for primary execution unit.
2. Secondary EU Mode, Key Size, Key, and Data Size registers are configured as appropriate.
3. Hash-only Header: *L1* bytes fetched from address *P1* and written to secondary EU input FIFO.

4. Cipher Context: *L3* bytes are fetched from *P3* and written into the primary EU context registers
5. Main Data In: *L4* bytes fetched from address *P4* and written to primary EU input FIFO. If Descriptor header indicates outbound, this data is also written to secondary EU input FIFO.
6. Main Data out: *L5* bytes fetched from primary EU output FIFO are written to address *P5*. If Descriptor header indicates inbound, this data is also written into the secondary EU input FIFO.
7. ICV Out: *L6* bytes of hash / HMAC / CRC result fetched from secondary EU and written to address *P6*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Hash-only Hdr Len	J1	--	-	Eptr1	P1: Address of Hash-only Header
Pointer 2	Cipher Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	Cipher Context Len	J3	--	-	Eptr3	P3: Address of Cipher Context
Pointer 4	Main Data Length	J4	--	-	Eptr4	P4: Address of Main Data In
Pointer 5	Main Data Length	J5	--	-	Eptr5	P5: Address of Main Data Out
Pointer 6	Computed ICV Len	J6	--	-	Eptr6	P6: Address of Computed ICV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-14. Descriptor Format hmac_snoop_no_afeu

As an example, take the first control word listed in the table below" 0x20031c22. The two rightmost nibbles, "22", indicate an inbound hmac_snoop_no_afeu descriptor without done notification. the two nibbles "1c" are written into MDEU mode register, which was selected on the basis of "3" appearing in the fourth nibble. The leftmost "2" indicates DESU for the primary execution unit, and the value "00" written into the DESU mode register.

Table 27-23. Descriptor Control Word examples for hmac_snoop_no_afeu Descriptors

control word (32 bit hex)	Description
20031c22	des-ecb hmac-sha-1 inbound
20031d22	des-ecb hmac-sha-256 inbound
20031e22	des-ecb hmac-md5 inbound
20031f22	des-ecb hmac-sha-224 inbound
200b1c22	des-ecb hmac-sha-384 inbound

Table 27-23. Descriptor Control Word examples for hmac_snoop_no_afeu Descriptors (Continued)

control word (32 bit hex)	Description
200b1e22	des-ecb hmac-sha-512 inbound
20131c20	des-ecb hmac-sha-1 outbound
20131d20	des-ecb hmac-sha-256 outbound
20131e20	des-ecb hmac-md5 outbound
20131f20	des-ecb hmac-sha-224 outbound
20231c22	triple-des-ecb hmac-sha-1 inbound
20231d22	triple-des-ecb hmac-sha-256 inbound
20231e22	triple-des-ecb hmac-md5 inbound
20231f22	triple-des-ecb hmac-sha-224 inbound
202b1c22	triple-des-ecb hmac-sha-384 inbound
202b1e22	triple-des-ecb hmac-sha-512 inbound
20331c20	triple-des-ecb hmac-sha-1 outbound
20331d20	triple-des-ecb hmac-sha-256 outbound
20331e20	triple-des-ecb hmac-md5 outbound
203b1c20	triple-des-ecb hmac-sha-384 outbound
203b1f20	triple-des-ecb hmac-sha-224 outbound
20431c22	des-cbc hmac-sha-1 inbound
20431d22	des-cbc hmac-sha-256 inbound
20431e22	des-cbc hmac-md5 inbound
204b1c22	des-cbc hmac-sha-384 inbound
204b1f22	des-cbc hmac-sha-224 inbound
20531c20	des-cbc hmac-sha-1 outbound
20531d20	des-cbc hmac-sha-256 outbound
20531e20	des-cbc hmac-md5 outbound
20531f20	des-cbc hmac-sha-224 outbound
205b1c20	des-cbc hmac-sha-384 outbound
205b1e20	des-cbc hmac-sha-512 outbound
205b1f20	des-cbc hmac-sha-224 outbound
20631c22	triple-des-cbc hmac-sha-1 inbound
20631d22	triple-des-cbc hmac-sha-256 inbound
20631e22	triple-des-cbc hmac-md5 inbound
20631f22	triple-des-cbc hmac-sha-224 inbound
206b1e22	triple-des-cbc hmac-sha-512 inbound
206b1f22	triple-des-cbc hmac-sha-224 inbound

Table 27-23. Descriptor Control Word examples for hmac_snoop_no_afeu Descriptors (Continued)

control word (32 bit hex)	Description
20731c20	triple-des-cbc hmac-sha-1 outbound
20731d20	triple-des-cbc hmac-sha-256 outbound
20731e20	triple-des-cbc hmac-md5 outbound
20731f20	triple-des-cbc hmac-sha-224 outbound
20831c22	des-cfb hmac-sha-1 inbound
20831f22	des-cfb hmac-sha-224 inbound
208b1e22	des-cfb hmac-sha-512 inbound
20931d20	des-cfb hmac-sha-256 outbound
209b1d20	des-cfb hmac-sha-256 outbound
209b1e20	des-cfb hmac-sha-512 outbound
20a31c22	triple-des-cfb hmac-sha-1 inbound
20a31d22	triple-des-cfb hmac-sha-256 inbound
20a31e22	triple-des-cfb hmac-md5 inbound
20ab1c22	triple-des-cfb hmac-sha-384 inbound
20b31e20	triple-des-cfb hmac-md5 outbound
20bb1c20	triple-des-cfb hmac-sha-384 outbound
20bb1f20	triple-des-cfb hmac-sha-224 outbound
20c31c22	des-ofb hmac-sha-1 inbound
20c31d22	des-ofb hmac-sha-256 inbound
20c31e22	des-ofb hmac-md5 inbound
20c31f22	des-ofb hmac-sha-224 inbound
20cb1c22	des-ofb hmac-sha-384 inbound
20cb1d22	des-ofb hmac-sha-256 inbound
20cb1e22	des-ofb hmac-sha-512 inbound
20cb1f22	des-ofb hmac-sha-224 inbound
20d31c20	des-ofb hmac-sha-1 outbound
20d31d20	des-ofb hmac-sha-256 outbound
20d31e20	des-ofb hmac-md5 outbound
20d31f20	des-ofb hmac-sha-224 outbound
20db1c20	des-ofb hmac-sha-384 outbound
20db1d20	des-ofb hmac-sha-256 outbound
20db1e20	des-ofb hmac-sha-512 outbound
20db1f20	des-ofb hmac-sha-224 outbound
20e31f22	triple-des-ofb hmac-sha-224 inbound

Table 27-23. Descriptor Control Word examples for hmac_snoop_no_afeu Descriptors (Continued)

control word (32 bit hex)	Description
20f31d20	triple-des-ofb hmac-sha-256 outbound
20f31e20	triple-des-ofb hmac-md5 outbound
20fb1c20	triple-des-ofb hmac-sha-384 outbound
20fb1e20	triple-des-ofb hmac-sha-512 outbound
60031c22	aes-ecb hmac-sha-1 inbound
60031d22	aes-ecb hmac-sha-256 inbound
60031e22	aes-ecb hmac-md5 inbound
60031f22	aes-ecb hmac-sha-224 inbound
600b1c22	aes-ecb hmac-sha-384 inbound
600b1f22	aes-ecb hmac-sha-224 inbound
60131c20	aes-ecb hmac-sha-1 outbound
60131d20	aes-ecb hmac-sha-256 outbound
60131e20	aes-ecb hmac-md5 outbound
60131f20	aes-ecb hmac-sha-224 outbound
601b1e20	aes-ecb hmac-sha-512 outbound
60231c22	aes-cbc hmac-sha-1 inbound
60231d22	aes-cbc hmac-sha-256 inbound
60231e22	aes-cbc hmac-md5 inbound
602b1d22	aes-cbc hmac-sha-256 inbound
602b1e22	aes-cbc hmac-sha-512 inbound
60331c20	aes-cbc hmac-sha-1 outbound
60331d20	aes-cbc hmac-sha-256 outbound
60331e20	aes-cbc hmac-md5 outbound
603b1d20	aes-cbc hmac-sha-256 outbound
603b1e20	aes-cbc hmac-sha-512 outbound
60431c22	aes-ofb hmac-sha-1 inbound
60431d22	aes-ofb hmac-sha-256 inbound
60431e22	aes-ofb hmac-md5 inbound
604b1e22	aes-ofb hmac-sha-512 inbound
604b1f22	aes-ofb hmac-sha-224 inbound
60531c20	aes-ofb hmac-sha-1 outbound
60531e20	aes-ofb hmac-md5 outbound
605b1d20	aes-ofb hmac-sha-256 outbound
622b1c22	aes-cbc-rbp hmac-sha-384 inbound

Table 27-23. Descriptor Control Word examples for hmac_snoop_no_afeu Descriptors (Continued)

control word (32 bit hex)	Description
62331f20	aes-cbc-rbp hmac-sha-224 outbound
642b1c22	aes-xts hmac-sha-384 inbound
643b1c20	aes-xts hmac-sha-384 outbound
68631e22	aes-cfb hmac-md5 inbound
686b1d22	aes-cfb hmac-sha-256 inbound
68731f20	aes-cfb hmac-sha-224 outbound
687b1e20	aes-cfb hmac-sha-512 outbound
20080022	des-ecb crc32-IEEE-802 inbound
20080122	des-ecb crc32-IETF-3385 inbound
20080322	des-ecb crc32-custom inbound
20180020	des-ecb crc32-IEEE-802 outbound
20180120	des-ecb crc32-IETF-3385 outbound
20280122	triple-des-ecb crc32-IETF-3385 inbound
20380020	triple-des-ecb crc32-IEEE-802 outbound
20380320	triple-des-ecb crc32-custom outbound
20480022	des-cbc crc32-IEEE-802 inbound
20480122	des-cbc crc32-IETF-3385 inbound
20580020	des-cbc crc32-IEEE-802 outbound
20580120	des-cbc crc32-IETF-3385 outbound
20580320	des-cbc crc32-custom outbound
20780020	triple-des-cbc crc32-IEEE-802 outbound
20880122	des-cfb crc32-IETF-3385 inbound
20980020	des-cfb crc32-IEEE-802 outbound
20980120	des-cfb crc32-IETF-3385 outbound
20980320	des-cfb crc32-custom outbound
20a80022	triple-des-cfb crc32-IEEE-802 inbound
20a80122	triple-des-cfb crc32-IETF-3385 inbound
20a80322	triple-des-cfb crc32-custom inbound
20b80120	triple-des-cfb crc32-IETF-3385 outbound
20b80320	triple-des-cfb crc32-custom outbound
20c80022	des-ofb crc32-IEEE-802 inbound
20c80122	des-ofb crc32-IETF-3385 inbound
20d80020	des-ofb crc32-IEEE-802 outbound
20d80320	des-ofb crc32-custom outbound

Table 27-23. Descriptor Control Word examples for hmac_snoop_no_afeu Descriptors (Continued)

control word (32 bit hex)	Description
20e80022	triple-des-ofb crc32-IEEE-802 inbound
20e80122	triple-des-ofb crc32-IETF-3385 inbound
20e80322	triple-des-ofb crc32-custom inbound
20f80020	triple-des-ofb crc32-IEEE-802 outbound
20f80320	triple-des-ofb crc32-custom outbound
60180020	aes-ecb crc32-IEEE-802 outbound
60180320	aes-ecb crc32-custom outbound
60280022	aes-cbc crc32-IEEE-802 inbound
60380020	aes-cbc crc32-IEEE-802 outbound
60380320	aes-cbc crc32-custom outbound
60480122	aes-ofb crc32-IETF-3385 inbound
60480322	aes-ofb crc32-custom inbound
60580320	aes-ofb crc32-custom outbound
62280322	aes-cbc-rbp crc32-custom inbound
62380020	aes-cbc-rbp crc32-IEEE-802 outbound
62380120	aes-cbc-rbp crc32-IETF-3385 outbound
62380320	aes-cbc-rbp crc32-custom outbound
64280022	aes-xts crc32-IEEE-802 inbound
64280122	aes-xts crc32-IETF-3385 inbound
64280322	aes-xts crc32-custom inbound
64380020	aes-xts crc32-IEEE-802 outbound
64380120	aes-xts crc32-IETF-3385 outbound
68780320	aes-cfb crc32-custom outbound

27.7.1.2.7 Descriptor Type 0101_0: common_nonsnoop_afeu

Descriptor type common_nonsnoop_afeu differs from descriptor type common_nonsnoop_no_afeu only in how the channel moves data internally. More specifically, the channel writes data found at address *P1* into the AFEU input FIFO, and reads data to be written into address *P5* from the AFEU output FIFO.

Table 27-24. common_nonsnoop_afeu Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0101_0 common_ nonsnoop_ afeu	Length	reserved	Context In (via In FIFO)	Cipher Key	Main Data In	Data Out	Context Out (via Out FIFO)	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Data Processing Steps (as managed by the Channel):

1. Setup: Mode, Key, Key Size and Data Size registers are configured as appropriate.
2. Context In: *L1* bytes are fetched from address *P1* and written into the AFEU input FIFO.
3. Data In: *L3* bytes are fetched from address *P3* and written into the AFEU input FIFO.
4. Data Out: read *L4* bytes from output FIFO and write to address *P4*.
5. Context Out: *L5* bytes read from AFEU output FIFO are written to address *P5*. (Descriptor control word must select context out)

Table 27-25. Descriptor Control word examples for common_nonsnoop_afeu Descriptors

control word (32 bit hex)	Description
10000050	arc-four confidentiality
10200050	arc-four confidentiality with s-box context out
10500050	arc-four confidentiality with reuse of previous s-box context
10700050	arc-four confidentiality with s-box context in and out

27.7.1.2.8 Descriptor Type 1000_0: pkeu_mm

pkeu_mm descriptors are designed for efficient use of PKEU when performing RSA computations, or for the occasional elliptic curve field operation. For elliptic curve point operations, refer to Section 27.7.1.2.20.

Table 27-26. Descriptor Format Summary for pkeu_mm

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1000_0 pkeu_mm	Length	“N” In	“B” In	“A” In	“E” In	“B” Out	reserved	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Data Processing Steps (as managed by the Channel):

1. Setup: Mode, Key Size and Data Size registers are configured as appropriate.
2. Modulus: *L0* bytes of modulus are fetched from address *P0* and written into the PKEU "N" register. The byte found at address *P0* is the most significant byte of the modulus, and the byte found at address *(P0 + L0 - 1)* is the least significant byte of the modulus
3. Secondary Operand: *L1* bytes are fetched from address *P1* and written into the PKEU "B" register. This secondary operand is used for the more simple arithmetic operations supported by PKEU, such as modular multiplication.
4. Primary Operand: *L2* bytes are fetched from address *P2* and written into the PKEU "A" register. This primary operand is required for most operations. When computing modular exponentiation, the base is the primary operand.
5. Exponent: *L3* bytes are moved from address *P3* to the PKEU "E" register. For most operations, the exponent field is unused; it is the exponent for modular exponentiation. In addition, when computing R_nR_p mod P (conversion of modulus from N to P), this field is used to provide the *size* of the modulus N -- *L3* must be programmed with the size of the original modulus N. Because the original modulus N itself is not required, *P3* should be zero.
6. Result: *L4* bytes are read from the PKEU "B" register and written to address *P4*.

Table 27-27. Descriptor Control word examples for pkeu_mm Descriptors

control word (32 bit hex)	Description
50200080	modular exponentiation -- A^E mod N
50300080	R^2 mod N (F_p) computation
50400080	R_nR_P mod P (F_p) computation
50d00080	R^2 mod N (F_2m) computation
50e00080	inverse (F_2m) computation
50f00080	modular inverse (F_p) computation
51000080	modular addition computation
52000080	modular subtraction computation
53000080	modular multiplication without montgomery reduction
54000080	modular multiplication with montgomery reduction
55000080	binary field (F_2m) addition computation
56000080	binary field (F_2m) multiplication without montgomery reduction
57000080	binary field (F_2m) multiplication with montgomery reduction
58000080	rsa-single step computation

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Modulus Length	J0	--	-	Eptr0	P0: Address of Modulus
Pointer 1	Second Operand Len	J1	--	-	Eptr1	P1: Address of Second Operand
Pointer 2	First Operand Length	J2	--	-	Eptr2	P2: Address of First Operand
Pointer 3	Exponent Length	J3	--	-	Eptr3	P3: Address of Exponent
Pointer 4	Result Length	J4	--	-	Eptr4	P4: Address of computation result destination
Pointer 5	--	J5	--	-	Eptr5	P5: --
Pointer 6	--	J6	--	-	Eptr6	P6: --
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-15. Descriptor Format pkeu_mm

27.7.1.2.9 Descriptor Type 1100_0: hmac_snoop_aesu_ctr

Table 27-28. hmac_snoop_aesu_ctr Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1100_0 hmac_snoop_aesu_ctr	Length	Hash Key	Hash-only Header	AES Key	AES Context In	Main Data In	Data Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Descriptor type hmac_snoop_aesu_ctr is designed for use with AESU for Counter mode operation. For hmac_snoop_aesu_ctr, the Channel accesses AESU Context Registers differently than most other descriptor types, permitting the transfer of a smaller AESU context when required. Descriptor type hmac_snoop_aesu_ctr provides for snooping ciphertext to a secondary EU (either CRCU or MDEU).

Data Processing Steps for descriptor type hmac_snoop_aesu_ctr (as managed by the Channel):

1. AESU Mode, Key Size, Key, and Data Size registers are configured as appropriate.
2. Secondary EU Mode, Key Size, Key, and Data Size registers are configured as appropriate.
3. Hash-only Header: *L1* bytes fetched from address *P1* and written to secondary EU input FIFO.
4. Counter: driver builds Context in structure with initial counter value and counter modulus. Channel fetches this from address *P3* and writes into AESU Context registers.

5. Encrypt / Decrypt Input: *L4* bytes fetched from address *P4* and written to AESU input FIFO. If Descriptor header indicates outbound, this data is also written to secondary EU input FIFO.
6. Output: *L5* bytes fetched from AESU output FIFO are written to address *P5*. If Descriptor header indicates inbound, this data is also written into the secondary EU input FIFO.
7. Hash / HMAC / CRC: *L6* bytes fetched from secondary EU and written to address *P6*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Hash-only Hdr Len	J1	--	-	Eptr1	P1: Address of Hash-only Header
Pointer 2	AESU Key Length	J2	--	-	Eptr2	P2: Address of AESU Key
Pointer 3	AESU Context Len	J3	--	-	Eptr3	P3: Address of AESU Context
Pointer 4	Main Data Length	J4	--	-	Eptr4	P4: Address of Main Data In
Pointer 5	Main Data Length	J5	--	-	Eptr5	P5: Address of Main Data Out
Pointer 6	Computed ICV Len	J6	--	-	Eptr6	P6: Address of Computed ICV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-16. hmac_snoop_aesu_ctr Descriptor Format

Although the descriptor type is called "hmac_snoop", the descriptor type can be used for non-HMAC hashing, or for CRC generation (by specifying CRCU as the secondary EU).

Table 27-29. Descriptor Control word examples for hmac_snoop_aesu_ctr Descriptors

control word (32 bit hex)	Description
60631CC2	aes-ctr / hmac-sha-1 inbound
60631DC2	aes-ctr / hmac-sha-256 inbound
60631EC2	aes-ctr / hmac-md5 inbound
60631CC0	aes-ctr / hmac-sha-1 outbound
60631DC0	aes-ctr / hmac-sha-256 outbound
60631EC0	aes-ctr / hmac-md5 outbound
60631fc0	aes-ctr / hmac-sha-224 outbound
60631fc2	aes-ctr / hmac-sha-224 inbound
606b1cc0	aes-ctr / hmac-sha-384 outbound
606b1cc2	aes-ctr / hmac-sha-384 inbound
606b1dc0	aes-ctr / hmac-sha-256 outbound
606b1dc2	aes-ctr / hmac-sha-256 inbound
606b1ec0	aes-ctr / hmac-sha-512 outbound
606b1ec2	aes-ctr / hmac-sha-512 inbound
606b1fc0	aes-ctr / hmac-sha-224 outbound
606b1fc2	aes-ctr / hmac-sha-224 inbound

27.7.1.2.10 Descriptor Type 0000_1: IPsec_ESP

The IPsec_ESP descriptor type is designed to efficiently process IPsec ESP packets. It is suitable for most IPsec packets, but not for AES-GCM packets. For AES-GCM IPsec packets, use descriptors shown in Section 27.7.1.2.28 on page 159.

Table 27-30. IPsec ESP Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0000_1 ipsec_esp	Length	HMAC Key	Hash-only Header	Cipher IV In	Cipher Key	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	reserved	ICV In	ICV Out	reserved

27.7.1.2.11 IPsec-ESP Outbound

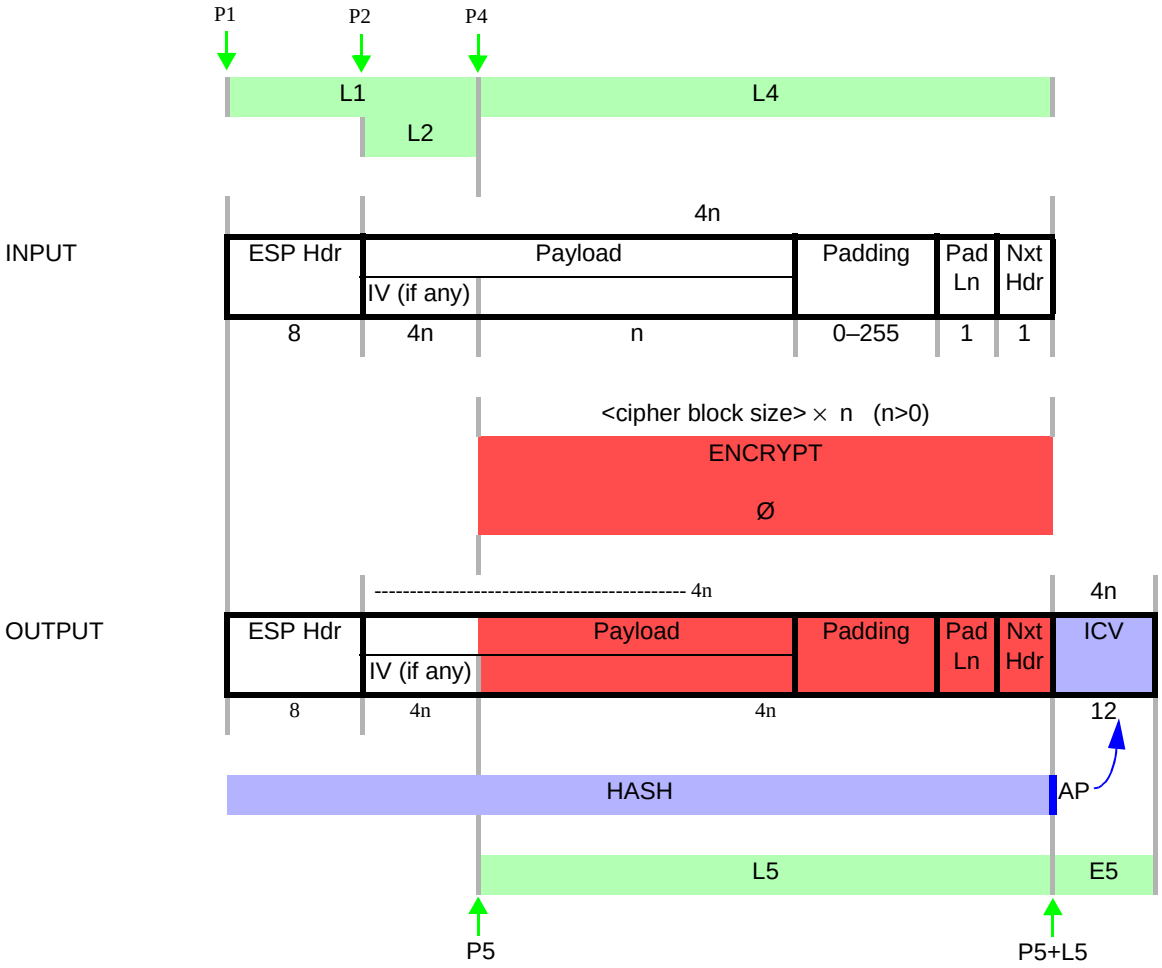


Figure 27-17. IPsec ESP Outbound Packet Pointer Diagram

Note: If the IV is in the packet (explicit IV), then the IV is the same as the last part of the hash-only header. In this case the L1 and L2 regions may overlap in memory, as shown above.

Data Processing Steps (as managed by the Channel):

1. Channel writes appropriate values to Mode Registers, Context, Key, Key Size and Data Size registers of selected Execution Units
2. Hash-only data (ESP HDR):
   - Starting at address *P1*, *L1* bytes fetched and fed to hashing EU input FIFO
   - SEC Channel computes hash-only datasize from *L1* and *L4*, and feeds to hashing EU Data Size register
3. Plaintext: Starting at *P4*, SEC fetches *L4* bytes and feeds them to the selected cipher EU input FIFO and to the selected hashing EU input FIFO.
4. Ciphertext: SEC fetches *L5* bytes from cipher EU output FIFO, and writes to *P5* and to hashing EU input FIFO.
5. ICV output: After the cipher EU has finished encryption, hashing EU finishes computation of the ICV in the MDEU. SEC obtains *E5* bytes of ICV from the hashing EU and writes to *P5*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Hash-only Hdr Len	J1	--	-	Eptr1	P1: Address of Hash-only Header
Pointer 2	Cipher IV Length	J2	--	-	Eptr2	P2: Address of Cipher IV
Pointer 3	Cipher Key Length	J3	--	-	Eptr3	P3: Address of Cipher Key
Pointer 4	Plaintext Length	J4	--	-	Eptr4	P4: Address of Plaintext
Pointer 5	Ciphertext Length	J5	ICV Len	-	Eptr5	P5: Address of Ciphertext • ICV Out
Pointer 6	Cipher IV Length	J6	--	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-18. IPsec ESP Outbound Descriptor Format

Below is a table containing selected Descriptor Header control words for descriptor type IPsec-ESP, outbound case. This is by no means an exhaustive list. Note that AES-GCM is not listed here because it is performed using the AES-GCM descriptor type (see Section 27.7.1.2.28).

Table 27-31. Descriptor Control word examples for IPsec ESP Outbound Descriptors

control word (32 bit hex)	Description
20531E08	DES-CBC / HMAC-MD5 Outbound
20512C08	DES-CBC / HMAC-SHA1 Outbound
20731E08	Triple-DES-CBC / HMAC-MD5 Outbound
20731C08	Triple-DES-CBC / HMAC-SHA1 Outbound
60331E08	AES-CBC / HMAC-MD5 Outbound
60331C08	AES-CBC / HMAC-SHA1 Outbound

27.7.1.2.12 IPsec-ESP Inbound

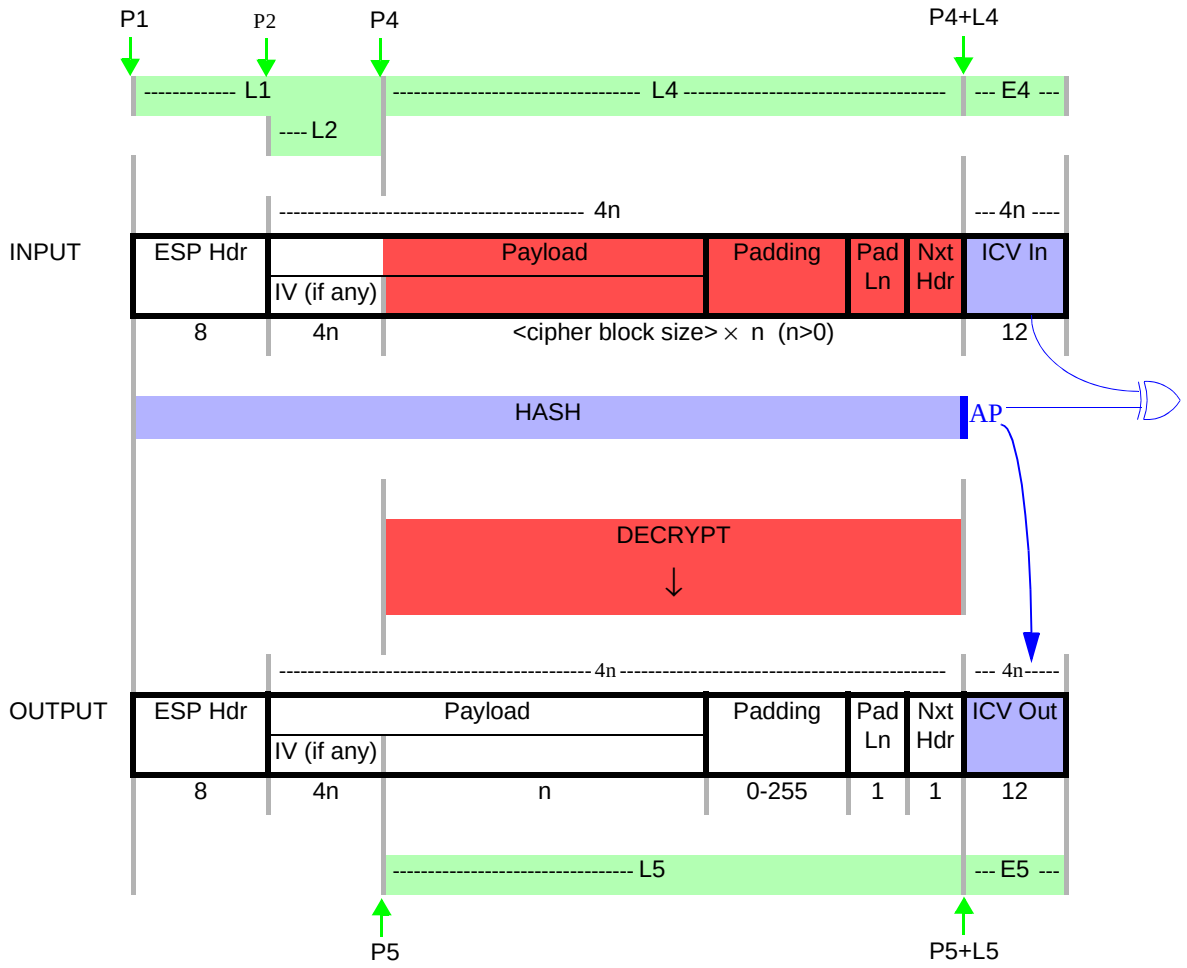


Figure 27-19. IPsec ESP Inbound Packet Pointer Diagram

Notes: **1.** If the IV is in the packet (explicit IV), then the IV is the same as the last part of the hash-only header. In this case the L1 and L2 regions may overlap in memory, as shown above.
2. When automatic ICV comparison is used, "ICV out" is typically not needed, so E5 may be set to 0.

Data Processing Steps (as managed by the Channel):

1. Hash-only: Starting at *P1*, read *L1* bytes and feed them to the MDEU.
2. Decrypt and hash: Starting at *P4*, read *L4* bytes and feed them to the cipher EU, with insnooping to the MDEU.
3. Output: Write *L5* bytes of cipher output data to *P5*.
4. ICV comparison: If ICV comparison is turned on, then continuing at *P4* read *E4* bytes and feed this "old ICV" to the MDEU. When the ICV computation in the MDEU is finished, compare the first *E4* bytes of MDEU output to the "old ICV", and send the pass/fail result back to the channel.

5. ICV output: Obtain *E5* bytes of ICV from the MDEU and write them continuing at *P5*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Hash-only Hdr Len	J1	--	-	Eptr1	P1: Address of Hash-only Header
Pointer 2	Cipher IV Length	J2	--	-	Eptr2	P2: Address of Cipher IV
Pointer 3	Cipher Key Length	J3	--	-	Eptr3	P3: Address of Cipher Key
Pointer 4	Plaintext Length	J4	ICV In Len	-	Eptr4	P4: Address of Plaintext • ICV In
Pointer 5	Ciphertext Length	J5	ICV Out Len	-	Eptr5	P5: Address of Ciphertext • ICV Out
Pointer 6	Cipher IV Length	J6	Extent6	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-20. IPsec ESP Inbound Descriptor Format

Note: If automatic ICV comparison is used, there is no need to output the new ICV. To avoid writing out this ICV, set *E5* to zero. On the other hand, if automatic ICV comparison is not used, then the new ICV should be written out. In this case it will be placed after the decrypted data. If the packet is being decrypted in place (i.e. if the same buffer is being used for packet data in and data out), then this would overwrite the original ICV, which would not be desirable, since the host needs to compare the old and new ICVs. The solution would be for the host to use the “scatter” capability to cause the new ICV to be written to some more convenient place.

Table 27-32 contains selected Descriptor Header control words for descriptor type IPsec-ESP, inbound case. This is by no means an exhaustive list. Note that AES-GCM is not listed here because it is performed using the AES-GCM descriptor type (see Section 27.7.1.2.28).

Table 27-32. Descriptor Control word examples for IPsec ESP Inbound Descriptors

control word (32 bit hex)	Description
20431C0A	DES-CBC / HMAC-SHA-1 Inbound
20431E0A	DES-CBC / HMAC-MD5 Inbound
20435C0A	DES-CBC / HMAC-SHA1 ICV Check Inbound
20435E0A	DES-CBC / HMAC-MD5 ICV Check Inbound
20631E0A	Triple-DES-CBC / HMAC-MD5 Inbound
20635E0A	Triple-DES-CBC / HMAC-MD5 ICV Check Inbound
20631C0A	Triple-DES-CBC / HMAC-SHA1 Inbound
20635C0A	Triple-DES-CBC / HMAC-SHA1 ICV Check Inbound
60231E0A	AES-CBC / HMAC-MD5 Inbound
60235E0A	AES-CBC / HMAC-MD5 ICV Check Inbound
60231C0A	AES-CBC / HMAC-SHA1 Inbound
60235C0A	AES-CBC / HMAC-SHA1 ICV Check Inbound

27.7.1.2.13 Descriptor Type 0001_1: IEEE 802.11i_aes_ccmp

Table 27-33. Descriptor Format Summary for IEEE 802.11i_aes_ccmp

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0001_1 802.11i AES ccmp	Length	CRC-only Header	AES Context In	AES Key	Hash-only Header	Main Data In	Data Out	AES Context Out
	Extent	CRC In/Out (FCS)	reserved	reserved	reserved	MIC In	MIC Out	reserved

27.7.1.2.14 IEEE 802.11i Outbound

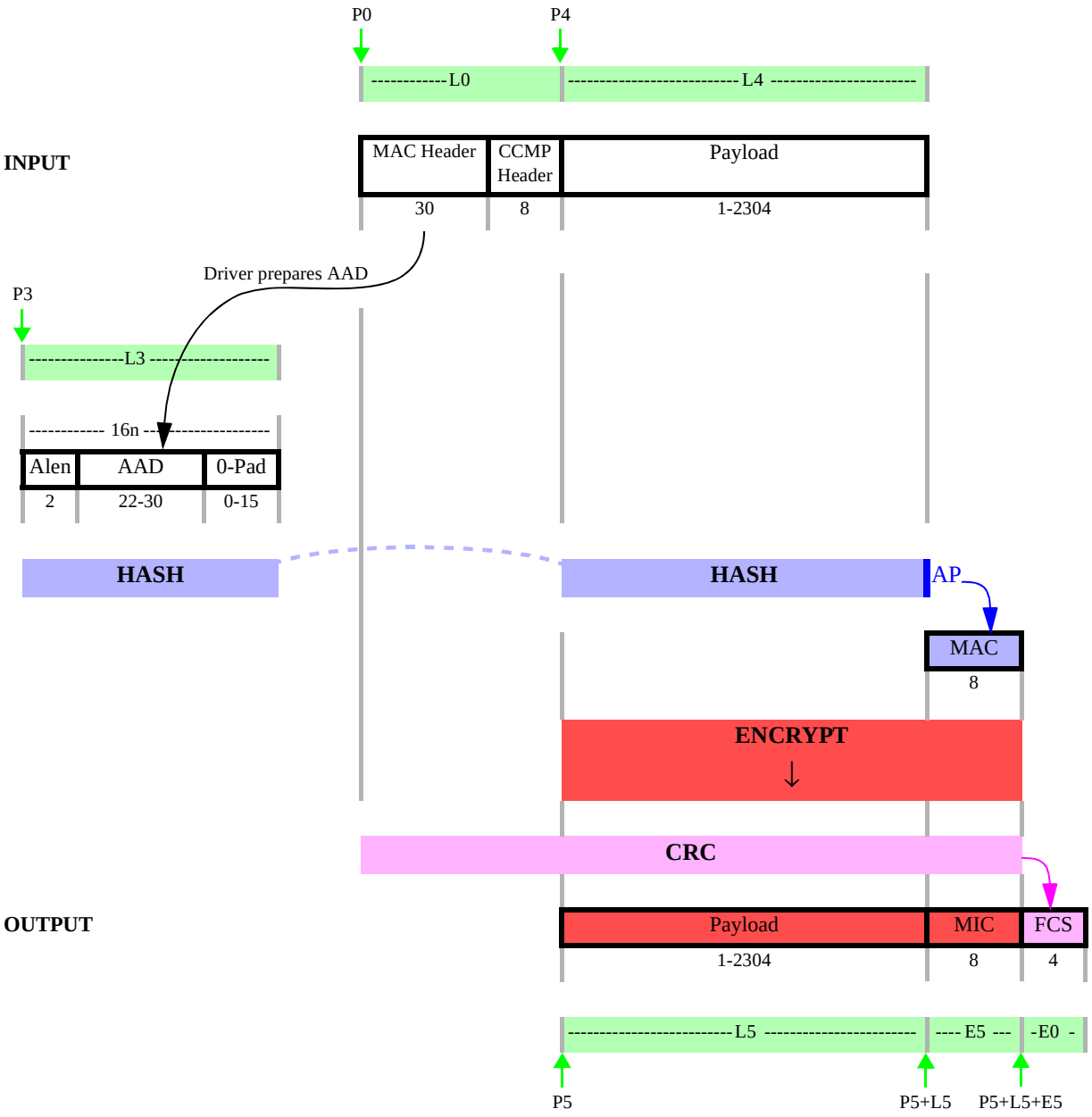


Figure 27-21. IEEE 802.11i (AES-CCM) Outbound Packet Pointer Diagram

Note: When implementing 802.11i CCMP, in preparation for authentication, the host must prepare the AAD field (additional authentication data) which is based on information in the MAC header. It must then add the Alen field (length of AAD), and zero-padding to make the hash-only length a multiple of 16B. The host must also supply the MAC Header and CCMP Header fields to accompany the encrypted output. 2.Hash autopadding is done with zeros.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	CRC Header Length	J0	FCS Len	-	Eptr0	P0: Address of CRC-Only Header Pointer
Pointer 1	AES Context Length	J1	--	-	Eptr1	P1: Address of AES Context In
Pointer 2	AES Key Length	J2	--	-	Eptr2	P2: Address of AES Key
Pointer 3	Length of AAD	J3	--	-	Eptr3	P3: AAD (Hash-only Data) Address
Pointer 4	Plaintext Length	J4	--	-	Eptr4	P4: Address of Plaintext
Pointer 5	Ciphertext Length	J5	ICV Len	-	Eptr5	P5: Address of Ciphertext • MIC Out • CRC Out
Pointer 6	AES Context Length	J6	--	-	Eptr6	P6: AES Context Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-22. IEEE 802.11 AES-CCMP Outbound Descriptor Format

Data Processing Steps (as managed by the Channel):

1. CRC Header: Starting at address *P0*, read *L0* bytes and send to CRCU input FIFO.
2. Hash-only: Set the AESU initialize bit. Starting at *P3*, read *L3* bytes and feed them to the AESU. The AESU automatically uses the *Alen* field to determine how much data is hash-only.
3. Encrypt and hash: Clear the AESU initialize bit. Starting at *P4*, read *L4* bytes, and feed them to the AESU. The AESU automatically does authentication and encryption on this region, and performs autopadding to complete the last block of data to the authentication function.
4. Output: Send *L5* bytes of cipher output data to *P5*. If CRCU is indicated in the descriptor header, this data is also sent to the CRCU input FIFO.
5. MIC output: Finish computation of the MAC in AESU, autopadding the data if necessary, and encrypt the result to form the MIC. Continuing at *P5*, write the MIC to memory. If CRCU is indicated in the descriptor header, the MIC is also sent to the CRCU input FIFO. The AESU also supplies the MIC as part of its Context output at *P6*.
6. CRCU: if CRCU is indicated by the descriptor header, get *E0* bytes of computed CRC from the CRCU, and write to memory continuing at *P5* (immediately after MIC).

Table 27-34. Descriptor Control word examples for CCMP Outbound Descriptors

control word (32 bit hex)	Description
6b100018	AES-CCM Outbound
6b180018	AES-CCM / CRC32-IEEE-802 Outbound

27.7.1.2.15 IEEE 802.11i Inbound

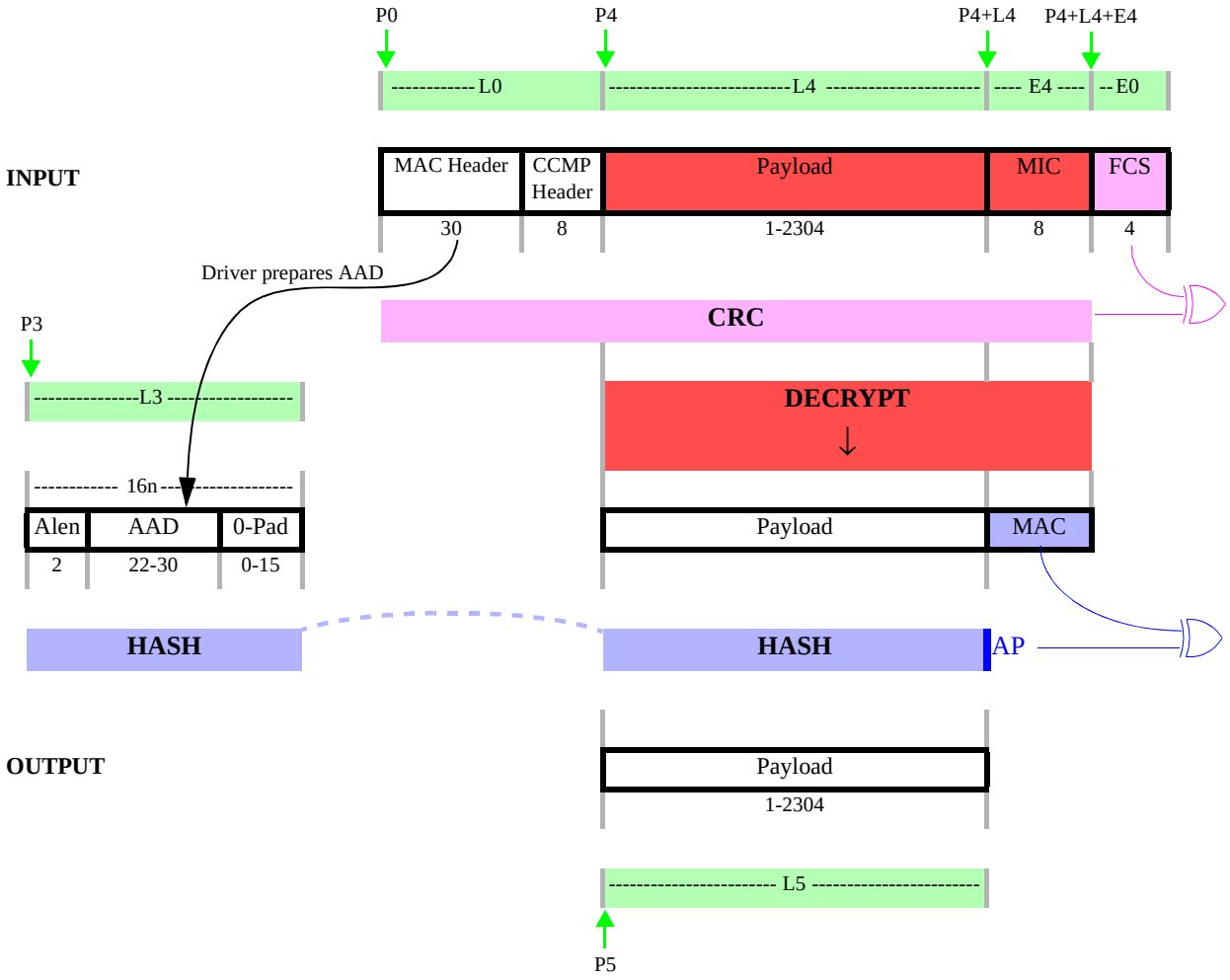


Figure 27-23. IEEE 802.11i (AES-CCM) Inbound Packet Pointer Diagram

Notes:
1. In preparation for authentication, the host must prepare the AAD field (additional authentication data) which is based on information in the MAC header. It must then add the Alen field (length of AAD), and zero-padding to make the hash-only length a multiple of 16B.
2. Hash autopadding is done with zeros.
3. When automatic ICV comparison is used, “Context out” is typically not needed, so L6 may be set to 0.

Data Processing Steps (as managed by the Channel):

1. CRC Header: Starting at address *P0*, read *L0* bytes and send to CRCU input FIFO.
2. Hash-only: Set the AESU initialize bit. Starting at *P3*, read *L3* bytes and feed them to the AESU (and the CRCU if the descriptor header specifies CRCU as a secondary EU). The AESU automatically uses the *Alen* field to determine how much data is hash-only.
3. Decrypt and hash: Clear the AESU initialize bit. Starting at *P4*, read *L4* bytes and feed them to the AESU (and the CRCU if the descriptor header specifies CRCU as a

secondary EU). The AESU automatically does authentication and decryption on this region, and performs autopadding to complete the last block for authentication.

4. MIC decryption: Continuing at *P4*, read *E4* bytes of old MIC and feed them to the AESU (and the CRCU if the descriptor header specifies CRCU as a secondary EU). The AESU decrypts this MIC to get the old MAC.
5. FCS: If *E0* is non-zero and the descriptor header specifies CRCU as a secondary EU, then continuing at *P4* read *E0* bytes of FCS and feed them to the CRCU input FIFO.
6. Output: Send *L5* bytes of cipher output data to *P5*.
7. ICV comparison: In the AESU, complete computation of the new MAC. If ICV comparison is turned on, compare the new MAC to the old MAC, and send the pass/fail result back to the channel.
8. FCS comparison: In the CRCU, complete computation of the CRC result. If CRC comparison is turned on, compare the computed CRC to the CRC residue.
9. ICV output: The computed MAC and decrypted MAC are part of the Context output at *P6*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	CRC Header Length	J0	FCS Len	-	Eptr0	P0: Address of CRC-Only Header Pointer
Pointer 1	AES Context Length	J1	--	-	Eptr1	P1: Address of AES Context In
Pointer 2	AES Key Length	J2	--	-	Eptr2	P2: Address of AES Key
Pointer 3	Length of AAD	J3	--	-	Eptr3	P3: AAD (Hash-only Data) Address
Pointer 4	Ciphertext Length	J4	MIC Len	-	Eptr4	P4: Address of Ciphertext • MAC In • FCS In
Pointer 5	Plaintext Length	J5	--	-	Eptr5	P5: Address of Plaintext
Pointer 6	AES Context Length	J6	--	-	Eptr6	P6: AES Context Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-24. IEEE 802.11 AES-CCMP Inbound Descriptor Format

For the IEEE 802.11 AES-CCMP descriptor type, the only valid execution units are AESU for primary EU, and CRCU for secondary EU. Any other selections are not recommended and may not perform identically in future SEC IP module.

Table 27-35. Descriptor Control word examples for CCMP Inbound Descriptors

control word (32 bit hex)	Description
6B00001A	AES-CCM Inbound
6F00001A	AES-CCM Inbound with ICV Check
6F08401A	AES-CCM with ICV Check / CRC32-IEEE-802 with Check

27.7.1.2.16 Descriptor Type 0010_1: SRTP

Table 27-36. SRTP Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
0010_1 srtp with ICV Check	Length	HMAC Key	AES Context In	AES Key	Main Data In	Data Out	HMAC Out	AES Context Out
	Extent	reserved	reserved	reserved	Hash-only Header	Hash-only Trailer	reserved	reserved
0010_1 srtp without ICV Check	Length	HMAC Key	AES Context In	AES Key	Main Data In	HMAC In	Data Out	AES Context Out
	Extent	reserved	reserved	reserved	Hash-only Header	Hash-only Trailer	HMAC Out	reserved

27.7.1.2.17 SRTP Outbound

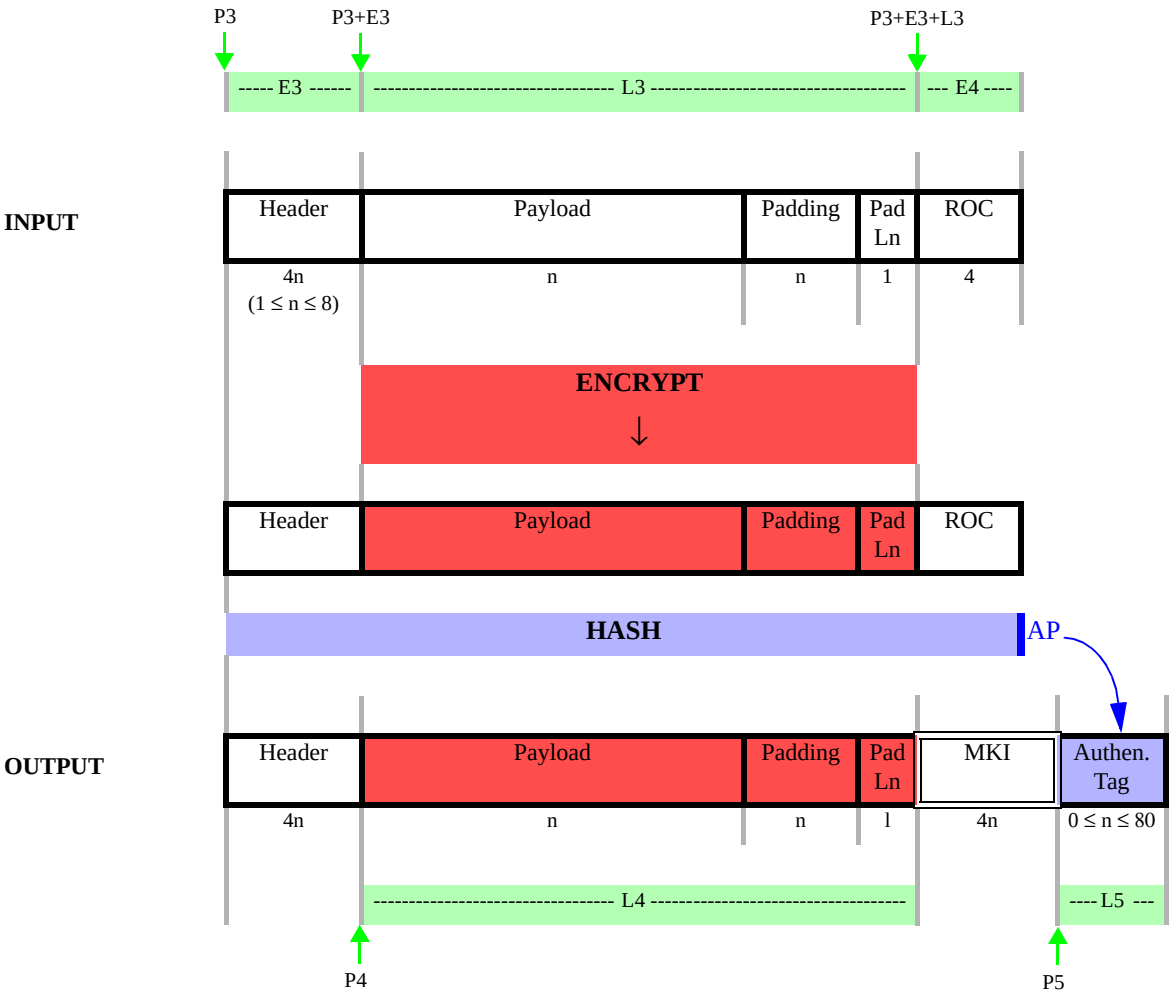


Figure 27-25. SRTP Outbound Packet Pointer Diagram

Data Processing Steps (as managed by the Channel):

1. Hash-only: Starting at *P3*, read *E3* bytes and feed them to the MDEU.
2. Encrypt and hash: Continuing at *P3*, read *L3* bytes and feed them to the cipher execution unit.
3. Output: Write *L4* bytes of cipher output to *P4*, with outsnooping to the MDEU.
4. Hash-only ROC: Continuing at *P3*, read *E4* bytes and feed them to the MDEU.
5. Authentication Tag output: Finish computation of the Authentication Tag in the MDEU. Obtain *L5* bytes of Authentication Tag from the MDEU and write them to *P5*.

Programming Notes:

- The ROC field must be contiguous with the Pad Ln field of the packet.

- Key length = 160b.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	AES Context Length	J1	--	-	Eptr1	P1: Address of Cipher Context In
Pointer 2	AES Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	Length of Plaintext	J3	Hdr Len	-	Eptr3	P3: Header • Plaintext • Trailer
Pointer 4	Ciphertext Length	J4	Trlr Len	-	Eptr4	P4: Address of Ciphertext
Pointer 5	AuthTag Length	J5	--	-	Eptr5	P5: Address of Authen. Tag Out
Pointer 6	Cipher Ctxt Length	J6	--	-	Eptr6	P6: Cipher Context Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-26. SRTP Outbound Descriptor Format

Table 27-37. Descriptor Control word example for SRTP Outbound Descriptors

control word (32 bit hex)	Description
64631C28	AES-CTR / HMAC SHA-1 SRTP Outbound

27.7.1.2.18 SRTP Inbound without ICV Compare

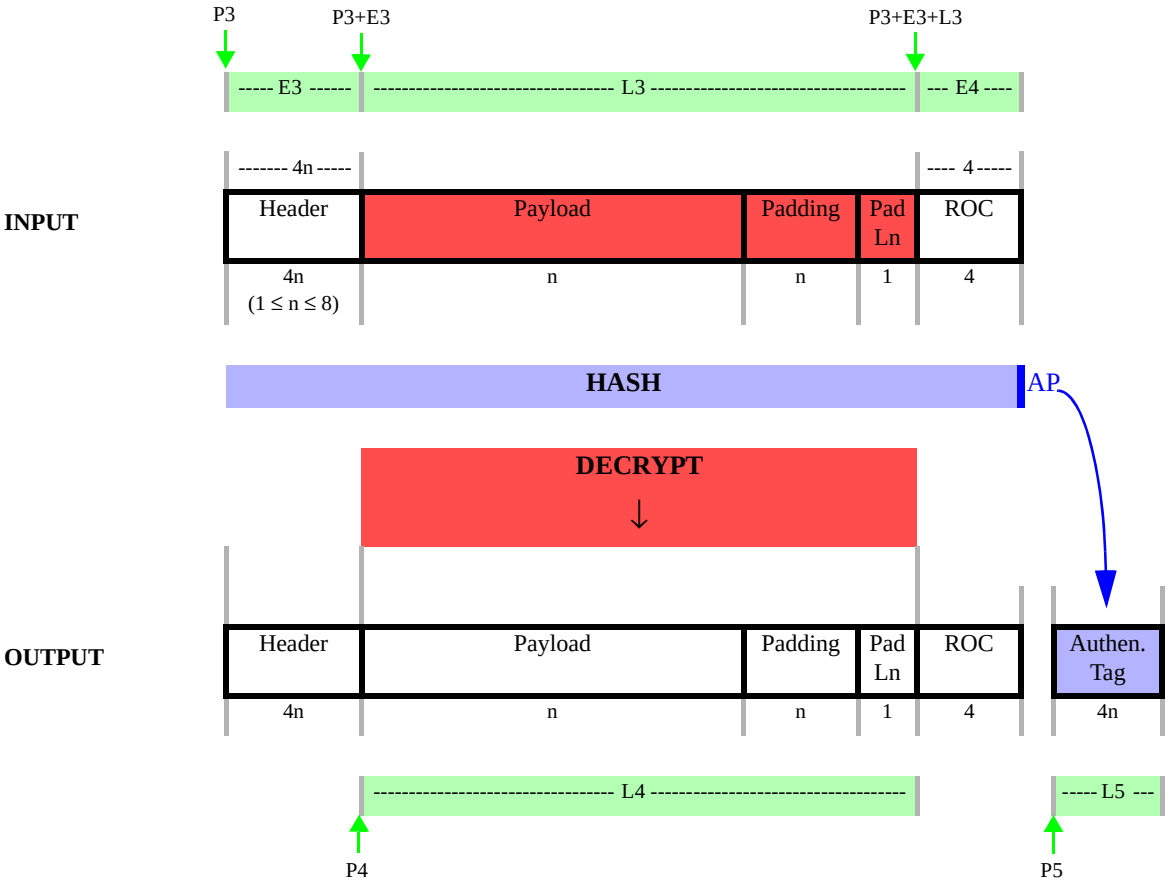


Figure 27-27. SRTP Inbound without ICV Compare Packet Pointer Diagram

Data Processing Steps (as managed by the Channel):

1. Hash-only: Starting at *P3*, read *E3* bytes and feed them to the MDEU.
2. Decrypt and hash: Continuing at *P3*, read *L3* bytes and feed them to the cipher EU, with insnooping to the MDEU.
3. Output: Write *L4* bytes of output data to *P4*.
4. Hash-only ROC: Continuing at *P3*, read *E4* bytes and feed them to the MDEU.
5. Authentication Tag: Finish computation of the Authentication Tag in the MDEU. Obtain *L5* bytes of Authentication Tag from the MDEU and write them to *P5*.

.

Programming Notes: See the notes for the outbound case.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	AES Context Length	J1	--	-	Eptr1	P1: Address of Cipher Context In
Pointer 2	AES Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	Length of Ciphertext	J3	Hdr Len	-	Eptr3	P3: Header • Ciphertext • Trailer
Pointer 4	Plaintext Length	J4	Trlr Len	-	Eptr4	P4: Address of Plaintext
Pointer 5	AuthTag Length	J5	--	-	Eptr5	P5: Address of Authen. Tag Out
Pointer 6	Cipher Ctxt Length	J6	--	-	Eptr6	P6: Cipher Context Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-28. SRTP Inbound without ICV Checking Descriptor Format

Table 27-38. Descriptor Control word example for SRTP Inbound Descriptor without ICV Checking

control word (32 bit hex)	Description
64631C2A	AES-CTR / HMAC SHA-1 SRTP Inbound

27.7.1.2.19 SRTP Inbound with ICV Compare

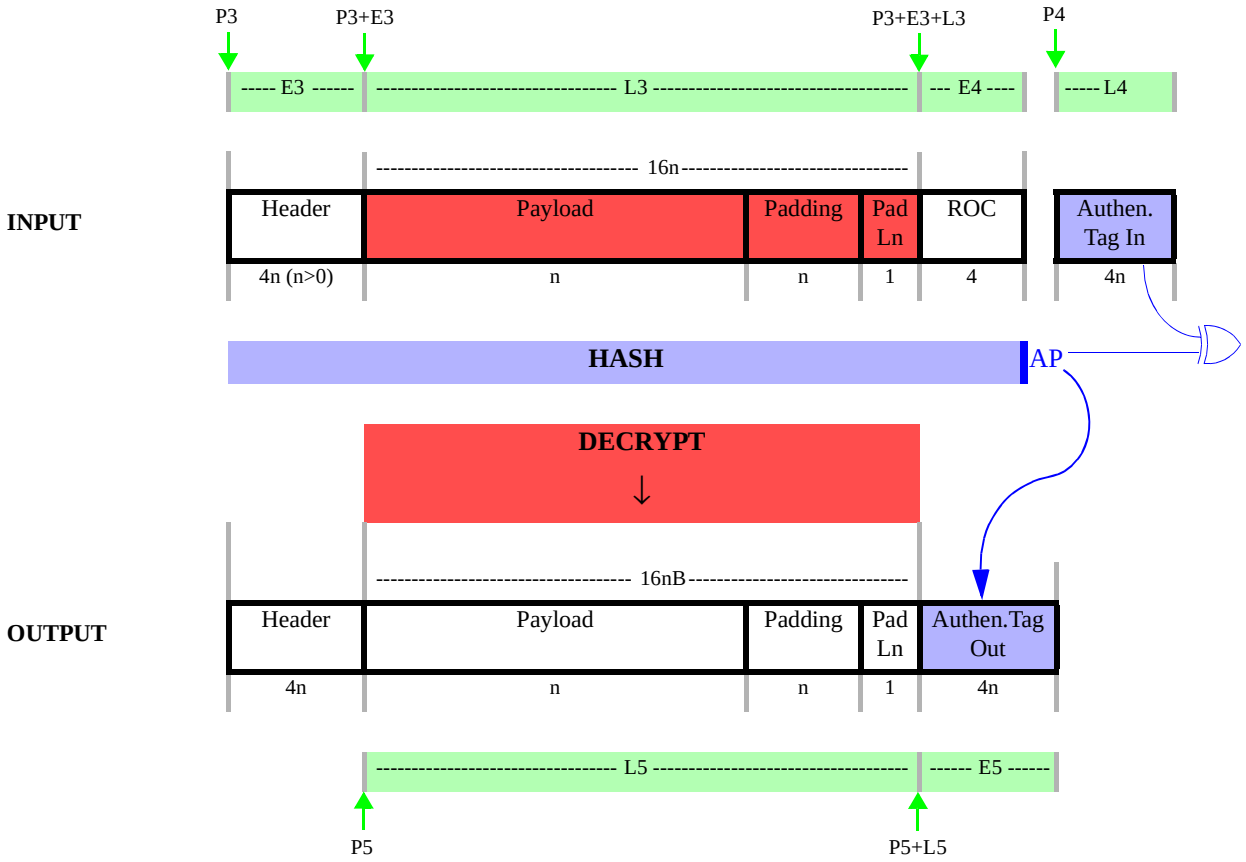


Figure 27-29. SRTP Inbound with ICV Compare Packet Pointer Diagram

Note: "Authentication tag out" is typically not needed, so E5 may be set to 0.

Data Processing Steps (as performed by the Channel):

1. Hash-only: Starting at *P3*, read *E3* bytes and feed them to the MDEU.
2. Decrypt and hash: Continuing at *P3*, read *L3* bytes and feed them to the cipher EU, with insnooping to the MDEU.
3. Output: Write *L5* bytes of output data to *P5*.
4. Hash-only ROC: Continuing at *P3*, read *E4* bytes and feed them to the MDEU.
5. ICV comparison: If ICV comparison is turned on, then starting at *P4* read *L4* bytes and feed this "old authentication tag" to the MDEU. When the ICV computation in the MDEU is finished, compare the first *L4* bytes of MDEU output to the "old authentication tag", and send the pass/fail result back to the channel.
6. ICV output: Obtain *E5* bytes of authentication tag from the MDEU and write them continuing at *P5*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	CRC Header Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	AES Context Length	J1	--	-	Eptr1	P1: Address of Cipher Context In
Pointer 2	AES Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	Length of Ciphertext	J3	Hdr Len	-	Eptr3	P3: Header • Ciphertext • Trailer
Pointer 4	AuthTag Length	J4	Trlr Len	-	Eptr4	P4: Address of Authen. Tag In
Pointer 5	Plaintext Length	J5	Tag Len	-	Eptr5	P5: Address of Plaintext • Authen. Tag Out
Pointer 6	Cipher Ctxt Length	J6	--	-	Eptr6	P6: Cipher Context Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-30. SRTP Inbound with ICV Checking Descriptor Format

Note: See the notes for the outbound case.

Table 27-39. Descriptor Control word examples for SRTP Inbound Descriptor with ICV Checking

control word (32 bit hex)	Description
64635C2A	AES-CTR / HMAC-SHA1 with ICV Check SRTP Inbound

27.7.1.2.20 Descriptor Types0011_1: pkeu_build, 0100_1: pkeu_ptmul, and 0101_1: pkeu_ptadd_dbl

These three descriptor types are defined for use with PKEU for elliptic curve cryptographic computations. pkeu_mm is summarized in 27-40 merely for comparative purposes.

Table 27-40. PKEU Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1000_0	Length	“N” In	“B” In	“A” In	“E” In	“B” Out	reserved	reserved
pkeu_mm	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0011_1	Length	“A0” In	“A1” In	“A2” In	“A3” In	“B0” In	“B1” In	"Build" Out
pkeu_build	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0100_1	Length	“N” In	“E” In	“Build” In	“B1” Out	“B2” Out	“B3” Out	reserved
pkeu_ptmul	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0101_1	Length	“N” In	“Build” In	“B2” In	“B3” In	“B1” Out	“B2” Out	“B3” Out
pkeu_ptadd_dbl	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

pkeu_build is a non-computational descriptor type defined for convenience in creating the parameter "Build" -- it simply takes the six parameters listed and creates a single 768-byte data structure to be loaded as a single structure in either the pkeu_ptmul or pkeu_ptadd_dbl descriptor. The same structure can easily be built by software or by using gather tables underneath either pkeu computational descriptor.

pkeu_build Data Processing Steps (as managed by the Channel):

1. Fetch *L0* bytes from address *P0* and write to PKEU "A0" register.
2. Fetch *L1* bytes from address *P1* and write to PKEU "A1" register.
3. Fetch *L2* bytes from address *P2* and write to PKEU "A2" register.
4. Fetch *L3* bytes from address *P3* and write to PKEU "A3" register.
5. Fetch *L4* bytes from address *P4* and write to PKEU "B0" register.
6. Fetch *L5* bytes from address *P5* and write to PKEU "B1" register.
7. Fetch *L6 bytes* from PKEU starting at register address "A0" and write to address *P6*. Note that for this to be useful, *L6* needs to specify at least 128 bytes for each non-zero-length segment in *L0* through *L4*. Further, any zero-length segments should be treated as non-zero for the sake of this computation. It is recommended that *L6* be 6*128 = 768 bytes.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	"A0" parameter Len	J0	--	-	Eptr0	P0: Address of data to put in pkeu A0 register
Pointer 1	"A1" parameter Len	J1	--	-	Eptr1	P1: Address of data to put in pkeu A1 register
Pointer 2	"A2" parameter Len	J2	--	-	Eptr2	P2: Address of data to put in pkeu A2 register
Pointer 3	"A3" parameter Len	J3	--	-	Eptr3	P3: Address of data to put in pkeu A3 register
Pointer 4	"B0" parameter Len	J4	--	-	Eptr4	P4: Address of data to put in pkeu B0 register
Pointer 5	"B1" parameter Len	J5	--	-	Eptr5	P5: Address of data to put in pkeu B1 register\
Pointer 6	"build" structure Len	J6	--	-	Eptr6	P6: Address of "build" structure
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-31. Descriptor Format pkeu_build

pkeu_ptmul is used to perform elliptic curve point multiplication.

pkeu_ptmul Data Processing Steps (as managed by the Channel):

1. Fetch *L0* bytes of modulus / irreducible polynomial from address *P0* and write to PKEU "N" register.
2. Fetch *L1* bytes of point multiplier from address *P1* and write to the PKEU "E" register.
3. Fetch *L2* bytes containing up to 6 elliptic curve parameters (at 128 bytes each) from address *P2* and write to PKEU starting at address for register "A0". If *L2* is 768 bytes, this will completely cover PKEU registers A0, A1, A2, A3, B0, and B1.
4. After computation has completed, fetch *L3* bytes of elliptic curve result x_3 and write to address *P3*.
5. After computation has completed, fetch *L4* bytes of elliptic curve result y_3 and write to address *P4*.
6. After computation has completed, fetch *L5* bytes of elliptic curve result z_3 and write to address *P5* (note that z_3 is of interest only if projective coordinates were used.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	"N" parameter Len	J0	--	-	Eptr0	P0: Address of modulus / irreducible for N register
Pointer 1	"E" parameter Len	J1	--	-	Eptr1	P1: Address of point multiplier for E register
Pointer 2	"Build" structure Len	J2	--	-	Eptr2	P2: Address of "build" structure
Pointer 3	"B1" result Len	J3	--	-	Eptr3	P3: Address for x_3 result
Pointer 4	"B2" result Len	J4	--	-	Eptr4	P4: Address for y_3 result
Pointer 5	"B3" result Len	J5	--	-	Eptr5	P5: Address for z_3 result
Pointer 6	--	J6	--	-	Eptr6	P6: --
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-32. Descriptor Format pkeu_ptmul

pkeu_ptadd_dbl is used to perform either elliptic curve point addition or elliptic curve point doubling.

pkeu_ptadd_dbl Data Processing Steps (as managed by the Channel):

1. Fetch *L0* bytes of modulus / irreducible polynomial from address *P0* and write to PKEU "N" register.
2. Fetch *L1* bytes containing up to 6 elliptic curve parameters (at 128 bytes each) from address *P1* and write to PKEU starting at address for register "A0". If *L1* is 768 bytes, this will completely cover PKEU registers A0, A1, A2, A3, B0, and B1.
3. Fetch *L2* bytes from address *P2* and write to PKEU "B2" register.
4. Fetch *L3* bytes from address *P3* and write to PKEU "B3" register.

5. After computation has completed, fetch *L4* bytes of elliptic curve result x_3 and write to address *P4*.
6. After computation has completed, fetch *L5* bytes of elliptic curve result y_3 and write to address *P5*.
7. After computation has completed, fetch *L6* bytes of elliptic curve result z_3 and write to address *P6* (note that z_3 is of interest only if projective coordinates were used.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	"N" parameter Len	J0	--	-	Eptr0	P0: Address of modulus / irreducible for N register
Pointer 1	"Build" structure Len	J2	--	-	Eptr2	P1: Address of "build" structure
Pointer 2	"B2" parameter Len	J2	--	-	Eptr2	P2: Address of data to put in pkeu B2 register
Pointer 3	"B3" parameter Len	J3	--	-	Eptr3	P3: Address of data to put in pkeu B3 register
Pointer 4	"B1" result Len	J3	--	-	Eptr3	P4: Address for x_3 result
Pointer 5	"B2" result Len	J4	--	-	Eptr4	P5: Address for y_3 result
Pointer 6	"B3" result Len	J5	--	-	Eptr5	P6: Address for z_3 result
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-33. Descriptor Format pkeu_ptadd_dbl

For elliptic curve computations operating on field elements rather than points, the pkeu_mm descriptor type is recommended.

Table 27-41. Descriptor Control word examples for elliptic-curve pkeu Descriptors

control word (32 bit hex)	Description
5ff00038	pkeu_assemble
50500048	pkeu F_p affine point multiply
50600048	pkeu F_2m affine point multiply
50700048	pkeu F_p projective point multiply
50800048	pkeu F_2m projective point multiply
50900058	pkeu F_p point addition
50a00058	pkeu F_2m point addition
50b00058	pkeu F_p point doubling
50c00058	pkeu F_2m point doubling

In 27-40 above, the identifiers listed are register names within PKEU. The table below is a cross-reference for *typical* use of those registers. For further information (such as exceptions to the table), see Section 27.6.1.11.

Table 27-42. Typical register usage for PKEU

PKEU Register	Typical EC Parameter	Description
A0	input: x_1 output: --	Input Point x coordinate
A1	input: y_1 output: --	Input point y coordinate
A2	input: z_1 output: --	Input Point z coordinate (byte string 0x01 if affine)
A3	input: a output: --	elliptic curve parameter "a"
B0	input: b / c output R^2 mod N	F_p: elliptic curve parameter "b" F_2m: elliptic curve parameter "c"; $= b^{(2^{m-2})} \bmod I$
B1	input (ptmul): R^2 mod N input (ptadd_dbl): x_2 output: x_3	input (ptmul): Montgomery conversion factor input (ptadd_dbl): second input point x coordinate output: Result point coordinate
B2	input (ptmul): -- input (ptadd_dbl): y_2 output: y_3	input (ptmul): unused input (ptadd_dbl): unused output: Result point coordinate
B3	input (ptmul): -- input (ptadd_dbl): z_2 output: z_3	input (ptmul): unused input (ptadd_dbl): second input point z coordinate output: Result point coordinate

For more information on PKEU operation and built in routines, see Section 27.6.1.11.

27.7.1.2.21 Descriptor Type 1000_1: tls_ssl_block

Table 27-43. tls_ssl_block Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1000_1 outbound tls_ssl_ block	Length	MAC Key	Cipher IV In	Cipher Key	Main Data In	reserved	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV Out	reserved	reserved
1000_1 inbound tls_ssl_ block	Length	MAC Key	Cipher IV In	Cipher Key	reserved	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV In	ICV Out	reserved

Descriptor type TLS / SSL block is used for TLS or SSL packets when a block cipher is chosen. As can be seen by comparing the packet pointer diagrams in **Figure 27-34** and **Figure 27-35**, the usage of a block cipher requires the insertion of a special padding field. The requirements for processing this special padding field results in different descriptor types for TLS / SSL processing block ciphers versus stream ciphers.

27.7.1.2.22 TLS / SSL Block Cipher Outbound

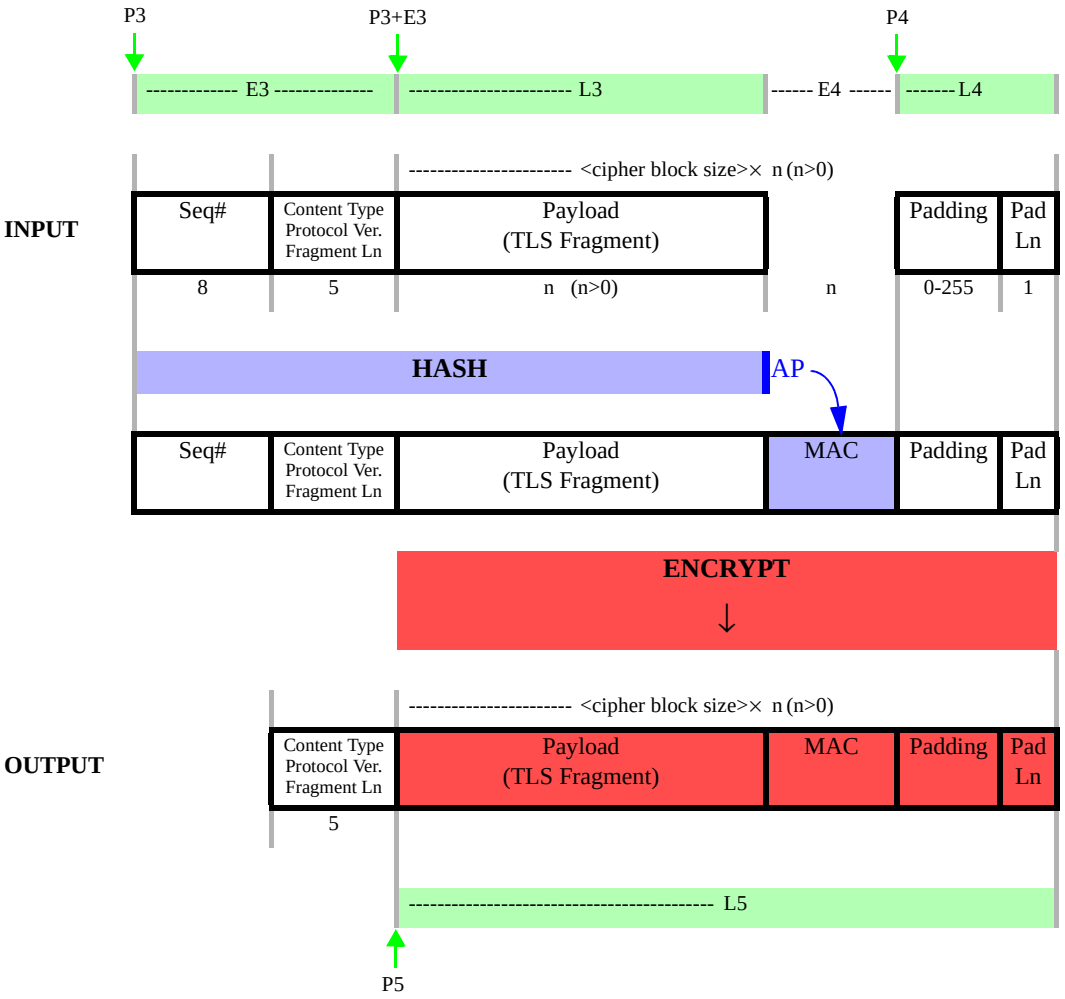


Figure 27-34. TLS / SSL Block Cipher Outbound Packet Pointer Diagram

Note: SSL has some different field lengths, but can be handled in the same manner.

Data Processing Steps (as managed by Channel)

1. Hash-only data: Starting at *P3*, read *E3* bytes and feed them to the MDEU.
2. Encrypt and hash: Continuing at *P3*, read *L3* bytes and feed them to the cipher EU, with insnooping to the MDEU.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Cipher IV Length	J1	--	-	Eptr1	P1: Address of Cipher IV In
Pointer 2	Cipher Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	Payload Length	J3	Hdr Len	-	Eptr3	P3: Address of Header • Payload
Pointer 4	Encrypt-only Length	J4	MAC Ln	-	Eptr4	P4: Address of Encrypt-only Padding
Pointer 5	Ciphertext Length	J5	--	-	Eptr5	P5: Address of Ciphertext Out
Pointer 6	Cipher IV Length	J6	--	-	Eptr6	P6: Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-35. TLS / SSL Block Cipher Outbound Descriptor Format

3. Output: Write cipher output data to *P5*.
4. MAC: Finish computation of the MAC in the MDEU. Continuing at P3, obtain *E4* bytes of MAC from the MDEU and direct them to the cipher EU.
5. Encrypt-only: Starting at *P4*, read *L4* bytes and feed them to the cipher EU. Continue writing cipher output data to *P5*.

Table 27-44. Descriptor Control word examples for TLS / SSL block cipher outbound Descriptors

control word (32 bit hex)	Description
20531888	des-cbc hmac-sha-1 outbound
20531988	des-cbc hmac-sha-256 outbound
20531a88	des-cbc hmac-md5 outbound
205b1888	des-cbc hmac-sha-384 outbound
205b1a88	des-cbc hmac-sha-512 outbound
20533088	des-cbc ssl-sha-1 outbound
20533288	des-cbc ssl-md5 outbound
20731888	triple-des-cbc hmac-sha-1 outbound
20731988	triple-des-cbc hmac-sha-256 outbound
20731a88	triple-des-cbc hmac-md5 outbound
20731b88	triple-des-cbc hmac-sha-224 outbound
207b1888	triple-des-cbc hmac-sha-384 outbound
207b1988	triple-des-cbc hmac-sha-256 outbound
207b1a88	triple-des-cbc hmac-sha-512 outbound
207b1b88	triple-des-cbc hmac-sha-224 outbound
20733088	triple-des-cbc ssl-sha-1 outbound

Table 27-44. Descriptor Control word examples for TLS / SSL block cipher outbound Descriptors (Continued)

control word (32 bit hex)	Description
20733288	triple-des-cbc ssl-md5 outbound
60331888	aes-cbc hmac-sha-1 outbound
60331988	aes-cbc hmac-sha-256 outbound
60331a88	aes-cbc hmac-md5 outbound
603b1988	aes-cbc hmac-sha-256 outbound
60333088	aes-cbc ssl-sha-1 outbound
60333288	aes-cbc ssl-md5 outbound

27.7.1.2.23 TLS / SSL Block Cipher Inbound

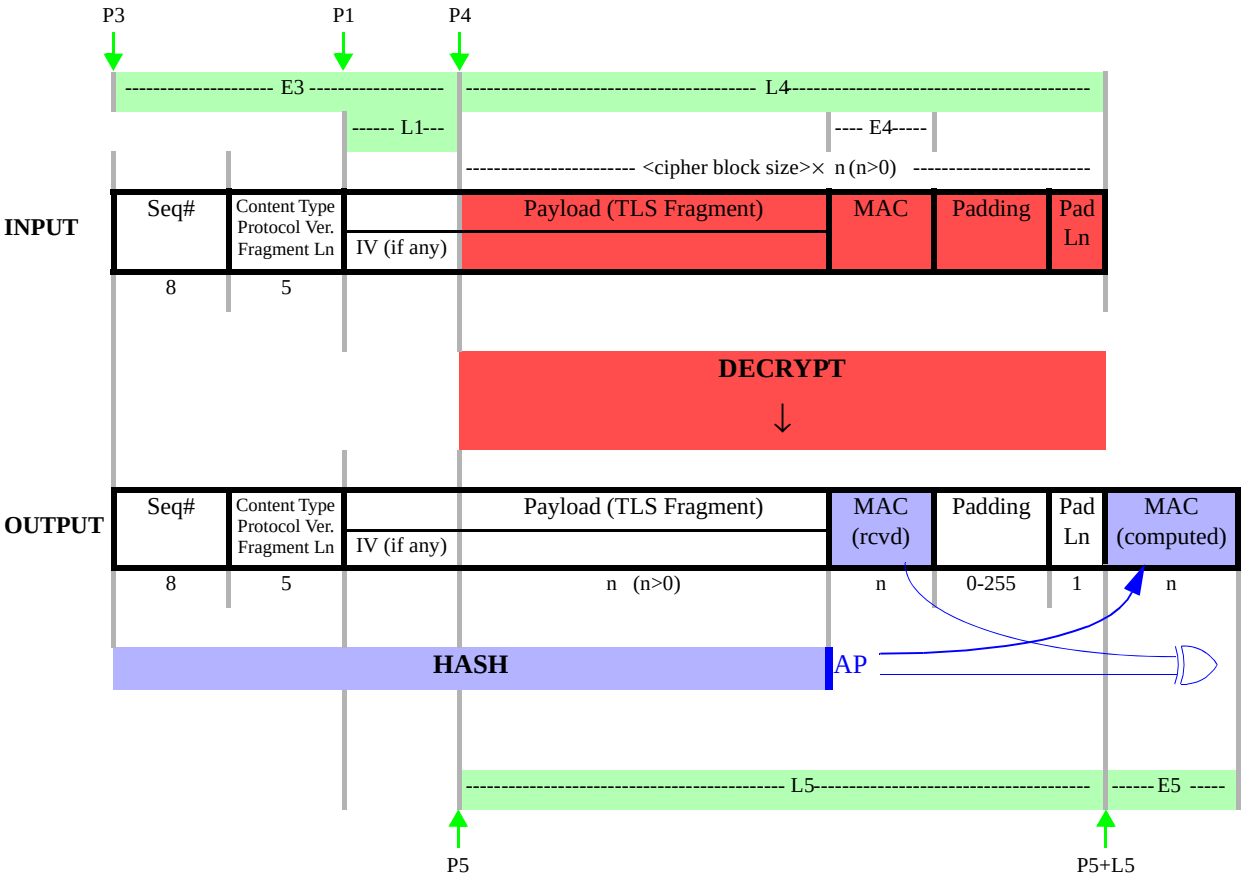


Figure 27-36. TLS / SSL Block Cipher Inbound Packet Pointer Diagram

Notes: **1.** SSL has some different field lengths, but can be handled in the same manner.
2. When automatic ICV comparison is used, “MAC out” is typically not needed, so E5 may be set to 0.
3. IV In occupies the initial L1 bytes of the payload for TLS 1.1/1.2.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Cipher IV Length	J1	--	-	Eptr1	P1: Address of Cipher IV In
Pointer 2	Cipher Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	--	J3	Hdr Len	-	Eptr3	P3: Address of Header
Pointer 4	Ciphertext Length	J4	MAC Ln	-	Eptr4	P4: Address of Ciphertext
Pointer 5	Payload Length	J5	MAC Ln	-	Eptr5	P5: Address of Payload (Plaintext) • MAC Out
Pointer 6	Cipher IV Length	J6	--	-	Eptr6	P6: Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-37. TLS / SSL Block Cipher Inbound Descriptor Format

Data Processing Steps (as managed by the Channel):

1. Write initial_data_size to the MDEU byte_count, where:

   initial_data_size = E3 + L4 - (256 + E4) bytes

2. Hash-only: Starting at *P3*, read *E3* bytes and feed them to the MDEU. As the MDEU processes data, it decrements its byte_count. If the byte_count is zero or negative, processing stops.
3. Decrypt and hash: Starting at *P4*, read *L4* bytes and feed them to the cipher EU with outsnooping to the MDEU. As the MDEU processes data, it decrements its byte_count. If the byte_count is zero or negative, processing stops.
4. Output: Write output data to *P5*.
5. Decrypt and hash (continued): The MDEU intercepts the last byte written to its input FIFO, which is the decrypted pad length. The MDEU computes:

   remaining_bytes = 255 - pad_length

   and adds this to its byte count. The MDEU continues processing data from its input FIFO until its byte count reaches zero.
6. MAC comparison: In the MDEU, complete computation of the MAC. Compare the newly computed MAC to the next *E4* bytes from the MDEU input FIFO, and send the pass/fail result back to the channel. Clear the MDEU input FIFO.
7. MAC output: Obtain *E5* bytes of MAC from the MDEU and write them continuing at *P5*.

Table 27-45. Descriptor Control word examples for TLS / SSL block cipher inbound Descriptors

control word (32 bit hex)	Description
2043188a	des-cbc hmac-sha-1 inbound
2043588a	des-cbc hmac-sha-1 icv check inbound
2043198a	des-cbc hmac-sha-256 inbound
2043598a	des-cbc hmac-sha-256 icv check inbound
20431a8a	des-cbc hmac-md5 inbound
20435a8a	des-cbc hmac-md5 icv check inbound
2043308a	des-cbc ssl-sha-1 inbound
2043708a	des-cbc ssl-sha-1 icv check inbound
2043328a	des-cbc ssl-md5 inbound
2043728a	des-cbc ssl-md5 icv check inbound
204b188a	des-cbc hmac-sha-384 inbound
204b588a	des-cbc hmac-sha-384 icv check inbound
2063188a	triple-des-cbc hmac-sha-1 inbound
2063588a	triple-des-cbc hmac-sha-1 icv check inbound
20631b8a	triple-des-cbc hmac-sha-224 inbound
20635b8a	triple-des-cbc hmac-sha-224 icv check inbound
2063198a	triple-des-cbc hmac-sha-256 inbound
2063598a	triple-des-cbc hmac-sha-256 icv check inbound
206b198a	triple-des-cbc hmac-sha-256 inbound
20631a8a	triple-des-cbc hmac-md5 inbound
20635a8a	triple-des-cbc hmac-md5 icv check inbound
2063308a	triple-des-cbc ssl-sha-1 inbound
2063708a	triple-des-cbc ssl-sha-1 icv check inbound
2063328a	triple-des-cbc ssl-md5 inbound
2063728a	triple-des-cbc ssl-md5 icv check inbound
206b598a	triple-des-cbc hmac-sha-256 icv check inbound
206b5a8a	triple-des-cbc hmac-sha-512 icv check inbound
6023188a	aes-cbc hmac-sha-1 inbound
6023588a	aes-cbc hmac-sha-1 icv check inbound
6023198a	aes-cbc hmac-sha-256 inbound
6023598a	aes-cbc hmac-sha-256 icv check inbound
60231a8a	aes-cbc hmac-md5 inbound
60235a8a	aes-cbc hmac-md5 icv check inbound

Table 27-45. Descriptor Control word examples for TLS / SSL block cipher inbound Descriptors (Continued)

control word (32 bit hex)	Description
6023308a	aes-cbc ssl-sha-1 inbound
6023708a	aes-cbc ssl-sha-1 icv check inbound
6023328a	aes-cbc ssl-md5 inbound
6023728a	aes-cbc ssl-md5 icv check inbound

27.7.1.2.24 Descriptor Type 1001_1: tls_ssl_stream

Table 27-46. Descriptor Format Summary for tls_ssl_stream

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1001_1 outbound tls_ssl_ stream	Length	MAC Key	Cipher IV In	Cipher Key	Main Data In	reserved	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV Out	reserved	reserved
1001_1 inbound tls_ssl_ stream	Length	MAC Key	Cipher IV In	Cipher Key	reserved	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV In	ICV Out	reserved

27.7.1.2.25 TLS / SSL Stream Cipher Outbound

Descriptor type TLS / SSL Stream is used for TLS or SSL packets when a stream cipher is chosen. As can be seen by comparing the packet pointer diagrams in **Figure 27-34** and <Cross Refs>Figure 27-35, the usage of a block cipher requires the insertion of a special padding field. The lack of a padding field allows for optimizations that result in different descriptor types for TLS / SSL processing stream ciphers versus block ciphers.

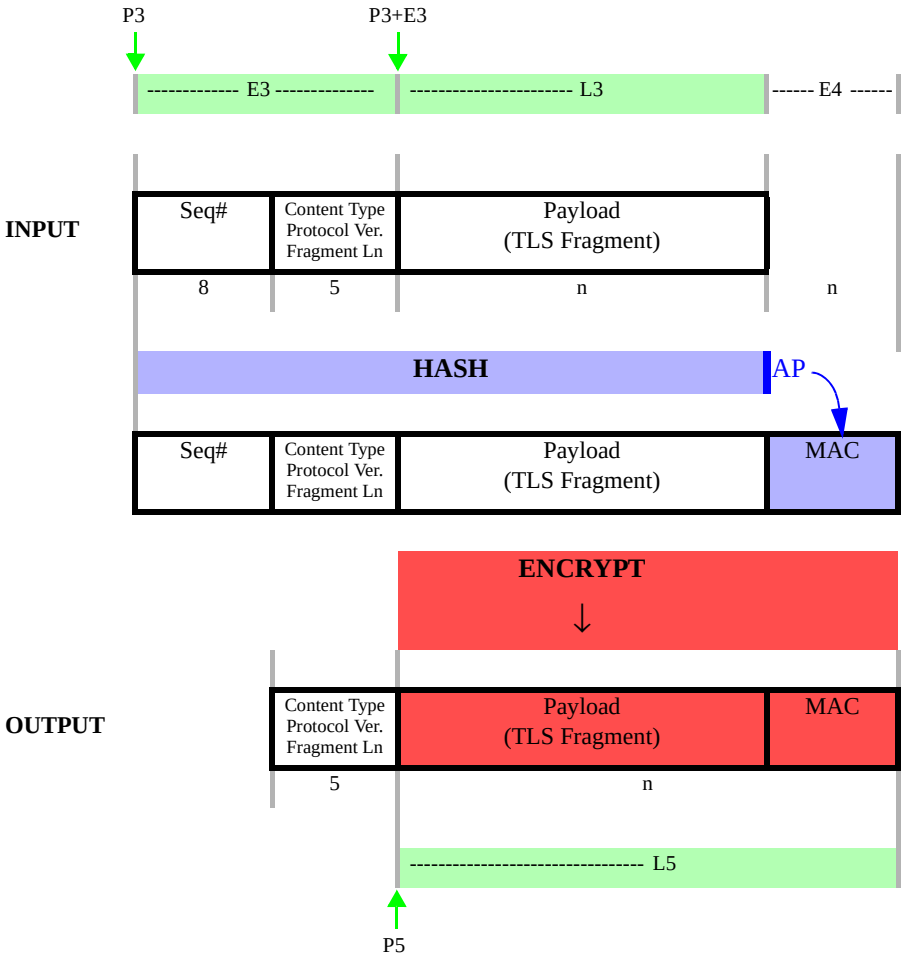


Figure 27-38. TLS / SSL Stream Cipher Outbound Packet Pointer Diagram

Note: SSL has some different field lengths, but can be handled in the same manner.

Data Processing Steps (as managed by the Channel); these are the same as the procedure for block cipher outbound except that there is no “encrypt only” padding at the end of the packet:

1. Hash-only: Starting at *P3*, read *E3* bytes and feed them to the MDEU.
2. Encrypt and hash: Continuing at *P3*, read *L3* bytes and feed them to the cipher EU, with insnooping to the MDEU.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Cipher IV Length	J1	--	-	Eptr1	P1: Address of Cipher IV In
Pointer 2	Cipher Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3	Payload Length	J3	Hdr Len	-	Eptr3	P3: Address of Header • Payload
Pointer 4	--	J4	MAC Ln	-	Eptr4	P4:--
Pointer 5	Ciphertext Length	J5	--	-	Eptr5	P5: Address of Ciphertext Out
Pointer 6	Cipher IV Length	J6	--	-	Eptr6	P6: Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-39. TLS / SSL Stream Cipher Outbound Descriptor Format

3. Output: Write output data to *P5*.
4. MAC: Finish computation of the MAC in the MDEU. Obtain *E4* bytes of MAC from the MDEU and direct them to the cipher EU.

Table 27-47. Descriptor Control word examples for TLS / SSL stream cipher outbound Descriptors

control word (32 bit hex)	Description
10031898	rc4 hmac-sha-1 outbound
10033098	rc4 ssl-sha-1 outbound
100b1898	rc4 hmac-sha-384 outbound
10231898	rc4 hmac-sha-1 outbound
10231998	rc4 hmac-sha-256 outbound
10231a98	rc4 hmac-md5 outbound
10233098	rc4 ssl-sha-1 outbound
10233298	rc4 ssl-md5 outbound
102b1898	rc4 hmac-sha-384 outbound
102b1a98	rc4 hmac-sha-512 outbound
10531898	rc4 hmac-sha-1 outbound
10531a98	rc4 hmac-md5 outbound
10533098	rc4 ssl-sha-1 outbound
105b1898	rc4 hmac-sha-384 outbound
105b1b98	rc4 hmac-sha-224 outbound
10731998	rc4 hmac-sha-256 outbound

Table 27-47. Descriptor Control word examples for TLS / SSL stream cipher outbound Descriptors (Continued)

control word (32 bit hex)	Description
10731a98	rc4 hmac-md5 outbound
10733098	rc4 ssl-sha-1 outbound
10733298	rc4 ssl-md5 outbound
107b1898	rc4 hmac-sha-384 outbound
107b1b98	rc4 hmac-sha-224 outbound

27.7.1.2.26 TLS / SSL Stream Cipher Inbound

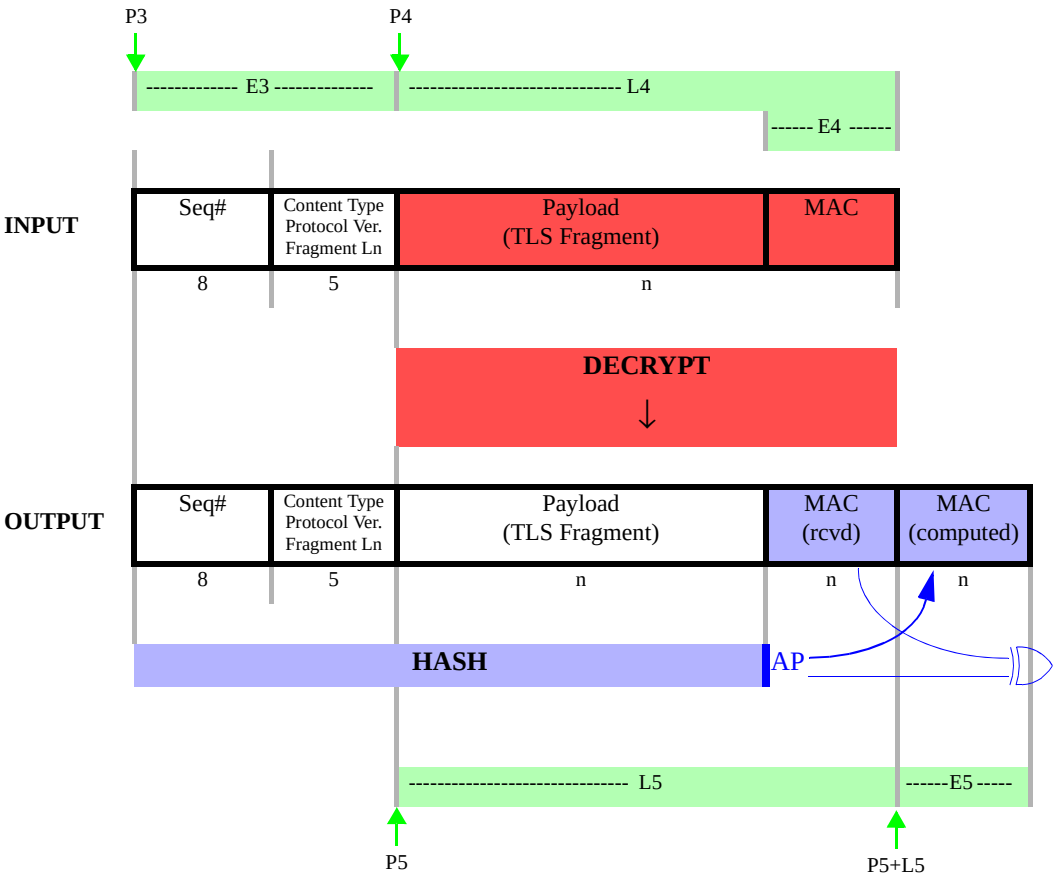


Figure 27-40. TLS / SSL Stream Cipher Inbound Packet Pointer Diagram

Notes: **1.** SSL has some different field lengths, but can be handled in the same manner.
2. When automatic ICV comparison is used, “MAC out” is typically not needed, so E5 may be set to 0.

Data Processing Steps (as managed by the Channel); note that these are very different from block cipher inbound:

1. Hash-only: Starting at *P3*, read *E3* bytes and feed them to the MDEU.
2. Decrypt and hash payload: Starting at *P4*, read *L4* bytes and feed them to the cipher EU, with outsnooping to the MDEU.
3. Output: Write output data to *P5*.
4. MAC comparison: In the MDEU, complete computation of the MAC. Compare the newly computed MAC to the last *E4* bytes from the MDEU input FIFO, and send the pass/fail result back to the channel. Clear the MDEU input FIFO.
5. MAC output: Obtain *E5* bytes of MAC from the MDEU and write them continuing at *P5*.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Hash Key Length	J0	--	-	Eptr0	P0: Address of Hash Key
Pointer 1	Cipher IV Length	J1	--	-	Eptr1	P1: Address of Cipher IV In
Pointer 2	Cipher Key Length	J2	--	-	Eptr2	P2: Address of Cipher Key
Pointer 3		J3	Hdr Len	-	Eptr3	P3: Address of Header
Pointer 4	Ciphertext Length	J4	MAC Ln	-	Eptr4	P4:Address of Ciphertext • MAC In
Pointer 5	Plaintext Length	J5	MAC Ln	-	Eptr5	P5: Address of Plaintext Out
Pointer 6	Cipher IV Length	J6	--	-	Eptr6	P6: Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-41. TLS / SSL Stream Cipher Inbound Descriptor Format

Table 27-48. Descriptor Control word examples for TLS / SSL stream cipher inbound Descriptors

control word (32 bit hex)	Description
1003189a	arc-four / hmac-sha-1 inbound
1003589a	arc-four / hmac-sha-1 icv check inbound
1003199a	arc-four / hmac-sha-256 inbound
1003599a	arc-four / hmac-sha-256 icv check inbound
100b589a	arc-four / hmac-sha-384 icv check inbound
1003309a	arc-four / ssl-sha-1 inbound
1003709a	arc-four / ssl-sha-1 icv check inbound
1003329a	arc-four / ssl-md5 inbound

Table 27-48. Descriptor Control word examples for TLS / SSL stream cipher inbound Descriptors (Continued)

control word (32 bit hex)	Description
1003729a	arc-four / ssl-md5 icv check inbound
1053589a	arc-four sbox in / hmac-sha-1 icv check inbound
1053199a	arc-four sbox in / hmac-sha-256 inbound
1053309a	arc-four sbox in / ssl-sha-1 inbound
1053709a	arc-four sbox in / ssl-sha-1 icv check inbound
1053329a	arc-four sbox in / ssl-md5 inbound
1053729a	arc-four sbox in / ssl-md5 icv check inbound
1073189a	arc-four sbox in and out / hmac-sha-1 inbound
1073599a	arc-four sbox in and out / hmac-sha-256 icv check inbound
10731a9a	arc-four sbox in and out / hmac-md5 inbound
10735a9a	arc-four sbox in and out / hmac-md5 icv check inbound
1073309a	arc-four sbox in and out / ssl-sha-1 inbound
1073329a	arc-four sbox in and out / ssl-md5 inbound
1073729a	arc-four sbox in and out / ssl-md5 icv check inbound
10235b9a	arc-four sbox out / hmac-sha-224 icv check inbound
1023199a	arc-four sbox out / hmac-sha-256 inbound
1023599a	arc-four sbox out / hmac-sha-256 icv check inbound
102b589a	arc-four sbox out / hmac-sha-384 icv check inbound
10231a9a	arc-four sbox out / hmac-md5 inbound
10235a9a	arc-four sbox out / hmac-md5 icv check inbound
1023309a	arc-four sbox out / ssl-sha-1 inbound
1023709a	arc-four sbox out / ssl-sha-1 icv check inbound
1023329a	arc-four sbox out / ssl-md5 inbound
1023729a	arc-four sbox out / ssl-md5 icv check inbound

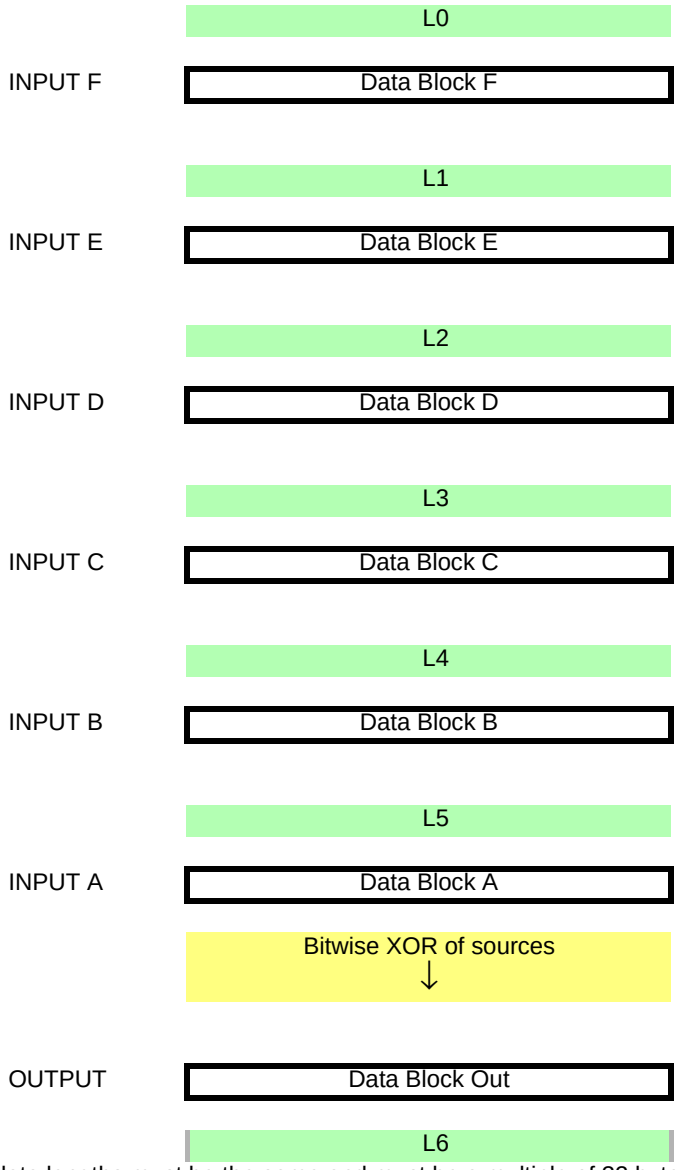
27.7.1.2.27 Descriptor Type 1010_1: raid_xor

The RAID-XOR descriptor type is designed for use when desired computation is an XOR of multiple sources into a single destination, such as for RAID level 5 data parity protection. Note that used pointers must be end-loaded for this descriptor type -- that is, all used pointer entries must be at the end and all unused pointers must be at the beginning of the descriptor. The channel will count the number of used pointers and write the appropriate value automatically into the SCM field of the AESU Mode register (**Figure 27-42**).

Table 27-49. Descriptor Format Summary for RAID-XOR

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1010_1 raid_xor	Length	Source F Data In	Source E Data In	Source D Data In	Source C Data In	Source B Data In	Source A Data In	Data Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

For this descriptor type, all lengths must be multiples of 32 bytes, and all J bits must be 0 (scatter-gather operation is not permitted for this descriptor type).



Notes: **1.** All data lengths must be the same and must be a multiple of 32 bytes.
2. Scatter/Gather is not allowed.

Figure 27-42. RAID-XOR Packet Pointer Diagram

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Data Block F Length	0	--	-	Eptr0	P0: Address of Data Block F
Pointer 1	Data Block E Length	0	--	-	Eptr1	P1: Address of Data Block E
Pointer 2	Data Block D Length	0	--	-	Eptr2	P2: Address of Data Block D
Pointer 3	Data Block C Length	0	--	-	Eptr3	P3: Address of Data Block C
Pointer 4	Data Block B Length	0	--	-	Eptr4	P4: Address of Data Block B
Pointer 5	Data Block A Length	0	--	-	Eptr5	P5: Address of Data Block A
Pointer 6	XOR Block Length	0	--	-	Eptr6	P6: Address of XOR Block Out
	Length	**J**[1]	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-43. RAID_XOR Descriptor Format

1. Jump bits are zero in the RAID_XOR descriptor format diagram because scatter and gather operation is not permitted.

Data Processing Steps (as managed by the Channel):

1. XOR: Read 32 bytes from *P5*, then *P4*, etcetera (for all consecutive non-zero pointers) and feed them to AESU. Continue reading 32-byte blocks in rotation from selected sources until *L0* bytes have been read from each.
2. Output: Write output data to *P6*.

Table 27-50. Descriptor Control word example for RAID-XOR Descriptors

control word (32 bit hex)	Description
6c6000a8	raid_xor

27.7.1.2.28 Descriptor Type 1011_1: aes_gcm

The AES_GCM descriptor type is overloaded to provide support for MACSec, IPsec, and any other AES-GCM needs. This descriptor is specific to the specific processing requirements of the AESU implementation of GCM.

Table 27-51. Descriptor Format Summary

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1011_1 ipsec_aes_gcm	Length	AES Context In	AAD In	Nonce Part 2 In	AES Key In	Main Data In	Data Out	Cipher Context Out
	Extent	reserved	reserved	reserved	Nonce Part 1 In	AES ICV In	AES ICV Out	CRC ICV In/Out

As can be seen by inspection of **Figure 27-46** versus **Figure 27-50**, the only difference between the two MACSec and IPsec descriptors is how the GCM-IV is provided to AESU. The Channel can handle any amount of GCM-IV that is provided as the last *E3* bytes of *P3*, and any amount of GCM-IV that is provided in *P2*. The portion of GCM-IV in *P3* is prepended to GCM-IV from *P2*. AES-GCM-IPsec descriptors are defined as AES-GCM descriptors with an E3 of 4. AES-GCM-MACSEC descriptors are defined as AES-GCM descriptors with an L2 of 0.

Table 27-52. Descriptor Control word examples for AES-GCM Outbound Descriptors

control word (32 bit hex)	Description
6a3000b8	aes gcm outbound
6a3800b8	aes gcm / crc-32 IEEE 802 outbound

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Context In Length	J0	--	-	Eptr0	P0: Address of AES-GCM Context
Pointer 1	GCM-AAD Length	J1	--	-	Eptr1	P1: Address of AES-GCM Additional Authen. Data
Pointer 2	Nonce Part 2 Length	J2	--	-	Eptr2	P2: Address of Part 2 of Nonce
Pointer 3	Key Length	J3	Nnc1 Ln	-	Eptr3	P3: Address of Cipher Key • Nonce Part 1 (Nnc1)
Pointer 4	Data In Length	J4	RICV Ln	-	Eptr4	P4: Address of Data In • Rcv ICV In • CRC In
Pointer 5	Data Out Length	J5	CICV Ln	-	Eptr5	P5: Address of Data Out • Comp ICV Out • CRC Out
Pointer 6	Cipher IV Length	J6	CRC Len	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Notes: **1.** Note that Rcv ICV In and CRC In are used only for inbound descriptor
2. Note that Comp ICV Out and CRC Out are used only for outbound descriptors

Figure 27-44. GCM General Descriptor Format

AES GCM Data Processing Steps[1] (as managed by the Channel):

1.this list of steps is true for General AES-GCM, for AES-GCM for MACSec, and AES-GCM for IPsec.

1. Channel prepares AESU Mode, Key Size, Key, Mac Size, and Data Size registers.
2. Channel fetches *E3* bytes from address *P3 + L3* and writes to AESU FIFO as first part of GCM-IV.
3. Channel fetches *L2* bytes from address *P2* and writes to AESU FIFO as remaining part of GCM-IV.
4. If *E3 + L2* is not a multiple of 16, the channel writes bytes of content zero to the AESU FIFO to pad the GCM-IV to a multiple of 16 bytes.[1]
5. Channel writes [*E3 + L2*] * 8 (length of IV in bits) to the AESU Context Registers len(IV)[c] field
6. Channel fetches *L1* bytes of AAD from address *P1* and writes to AESU FIFO (and the CRCU input FIFO if CRCU is specified in the header as a secondary EU)
7. If *L1* is not a multiple of 16, the channel writes bytes of content zero to the AESU FIFO (but not to the CRCU FIFO) to pad the AAD to a multiple of 16 bytes
8. Channel writes *L1* * 8 (length of AAD in bits) to the AESU Context Registers len(AAD)[c] field
9. If Pointer 4 specifies a receive ICV value, then it is written into the appropriate AESU Context registers.
10. *L4* bytes of Data In is fetched from address *P4* is written into the AESU input FIFO and to the CRCU input FIFO.
11. If *E6* is nonzero and the descriptor header specifies *inbound*, then write CRC to CRCU input FIFO.
12. Data Out is read from the AESU output FIFO and written to address *P5*.
13. If *E5* is nonzero, fetch the computed ICV and write to *P5 + L5*.
14. If *E6* is nonzero and the descriptor header specifies *outbound*, fetch the computed CRC and write to *P5 + L5 + E5*.

1.That the channel pads the GCM-IV and the AAD is pointed out only because the AESU section discusses the need for padding GCM-IV and AAD. Failure to note it here might leave the user with the mistaken belief that the padding is left to the host.

Table 27-53. Descriptor Control word examples for AES-GCM Inbound Descriptors

control word (32 bit hex)	Description
6a2000ba	aes gcm inbound
6a2800ba	aes gcm / crc-32 IEEE 802 inbound
6a2840ba	aes gcm / crc-32 IEEE 802 crc check inbound
6e2000ba	aes gcm icv check inbound
6e2800ba	aes gcm icv check / crc-32 IEEE 802 inbound
6e2840ba	aes gcm icv check / crc-32 IEEE 802 crc check inbound

Programming Notes:

1. CRC operations are optional.
2. The nonce, AAD, and payload are all sent to the AESU input FIFO. The channel pads the Nonce (GCM-IV) and AAD out to the next 16B word as written to AESU input FIFO, so that the AAD and payload start on a fresh block boundary.

27.7.1.2.29 AES_GCM Outbound for MACSec

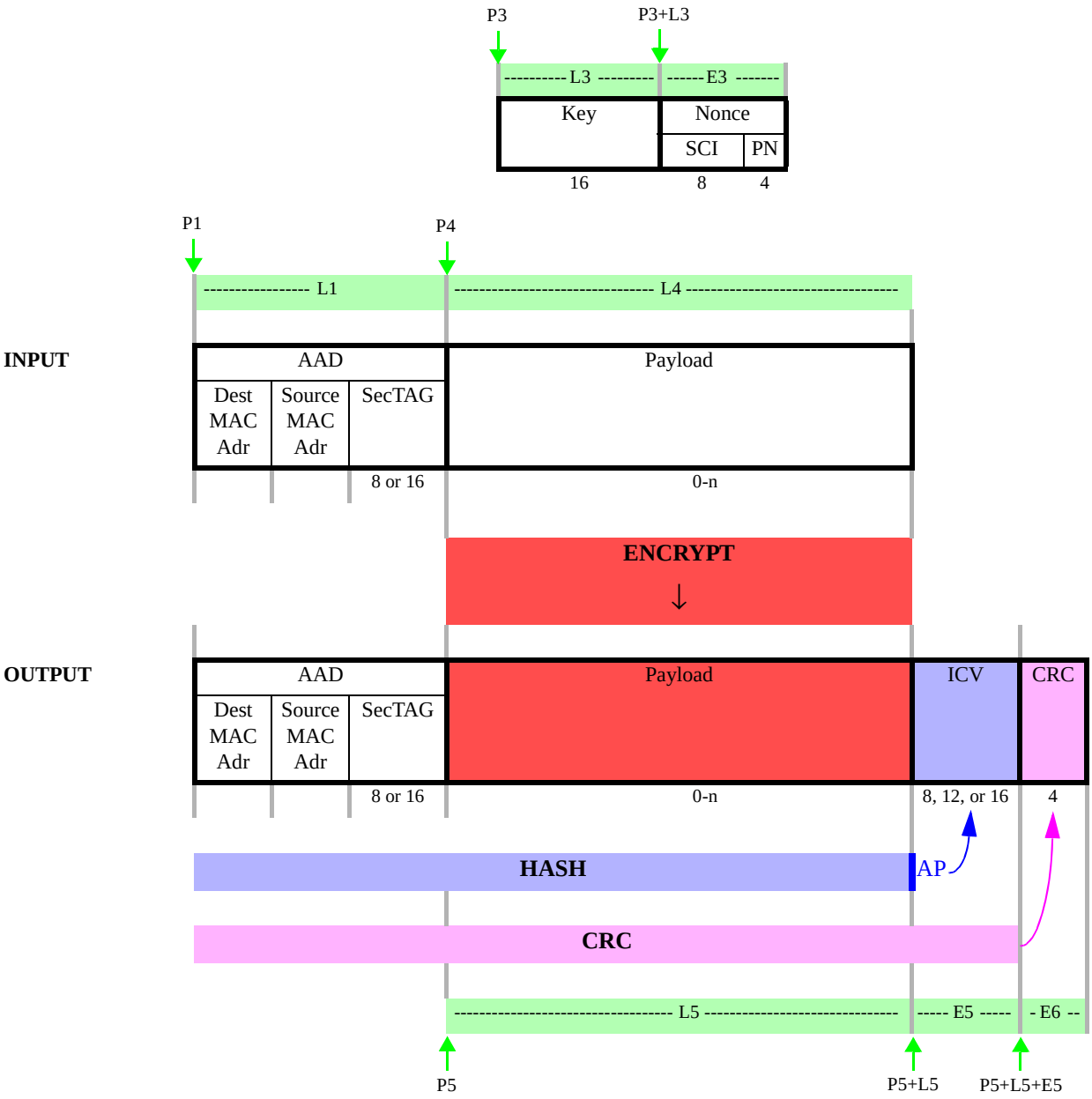


Figure 27-45. AES_GCM Outbound MACSec Packet Pointer Diagram

\Notes:

1. Driver must prepare AAD

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Context In Length	J0	--	-	Eptr0	P0: Address of AES-GCM Context
Pointer 1	GCM-AAD Length	J1	--	-	Eptr1	P1: Address of AES-GCM Additional Authen. Data
Pointer 2	--	J2	--	-	Eptr2	P2: (unused in this format)
Pointer 3	Key Length	J3	Nonce Ln	-	Eptr3	P3: Address of Cipher Key • Nonce
Pointer 4	Plaintext Length	J4	--	-	Eptr4	P4: Address of Plaintext
Pointer 5	Ciphertext Length	J5	ICV Len	-	Eptr5	P5: Address of Ciphertext • ICV Out • CRC Out
Pointer 6	Cipher IV Length	J6	CRC Len	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-46. GCM MACSec Outbound Descriptor Format

27.7.1.2.30 AES_GCM Inbound for MACSec

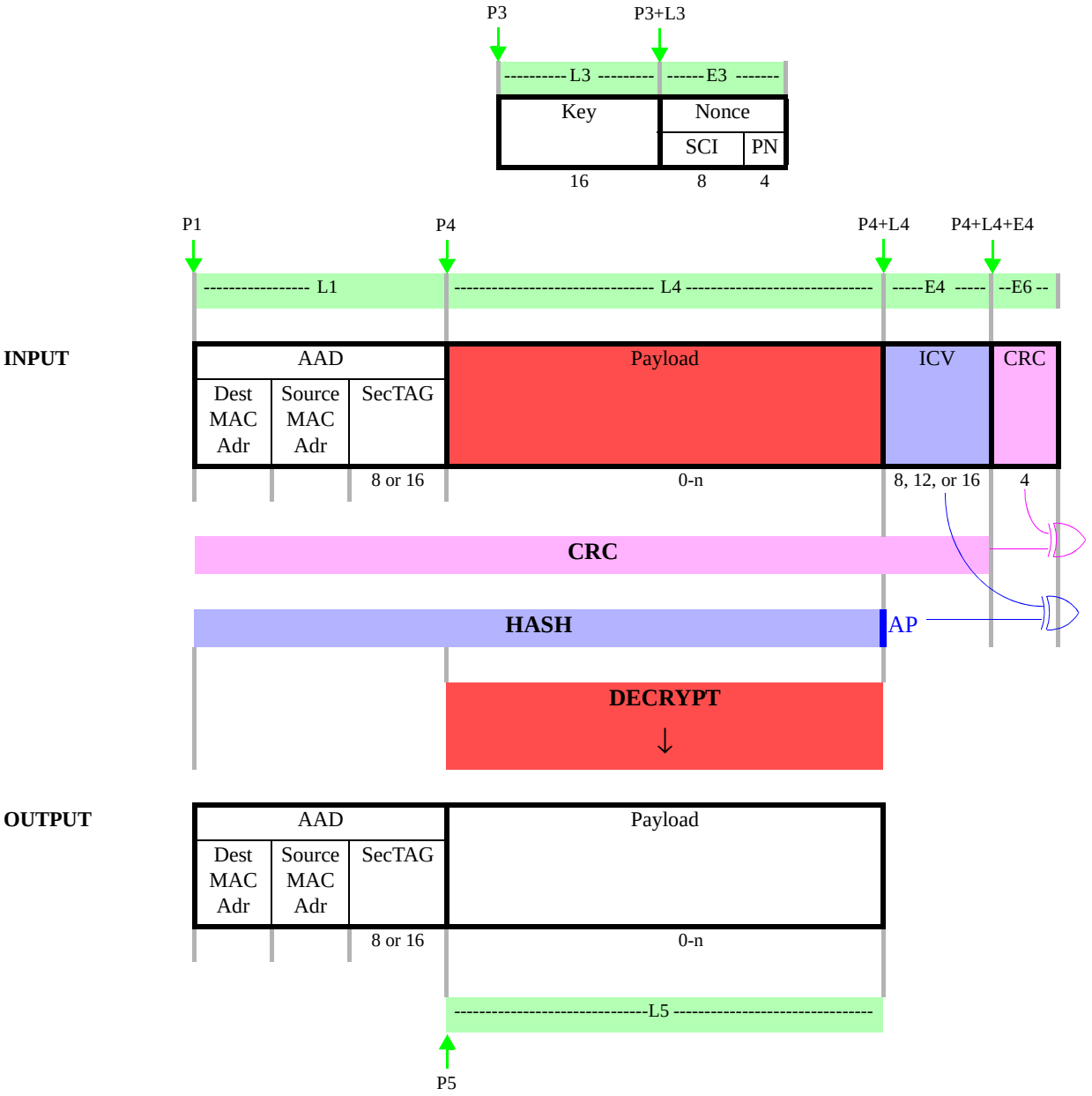


Figure 27-47. AES_GCM Inbound MACSec Packet Pointer Diagram

Notes:
1. CRC operations are optional. If CRC operation is used, then automatic CRC comparison must be activated. There is no CRC out.
2. When automatic ICV comparison is used for authentication, "ICV out" is typically not needed, so E5 may be set to 0.
3. The nonce, AAD, and payload are all sent to the AESU input FIFO. The Nonce and AAD must be padded out the next 16B word, so that the AAD and payload start on a fresh block boundary.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Context In Length	J0	--	-	Eptr0	P0: Address of AES-GCM Context
Pointer 1	GCM-AAD Length	J1	--	-	Eptr1	P1: Address of AES-GCM Additional Authen. Data
Pointer 2	--	J2	--	-	Eptr2	P2: (unused in this format)
Pointer 3	Key Length	J3	Nonce Ln	-	Eptr3	P3: Address of Cipher Key • Nonce
Pointer 4	Ciphertext Length	J4	ICV Len	-	Eptr4	P4: Address of Ciphertext In • ICV In • CRC In
Pointer 5	Plaintext Length	J5	--	-	Eptr5	P5: Address of Plaintext Out
Pointer 6	Cipher IV Length	J6	CRC Len	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-48. GCM MACSec Inbound Descriptor Format

27.7.1.2.31 AES_GCM Outbound for IPsec

To avoid a point of confusion, please note the term IV is overloaded for IPsec-AES-GCM. The IPsec-IV consists of SALT • GCM-IV. IPsec Salt for GCM comes from Key material, whereas IPsec-IV for GCM is transmitted (optionally) as part of the IPsec packet.

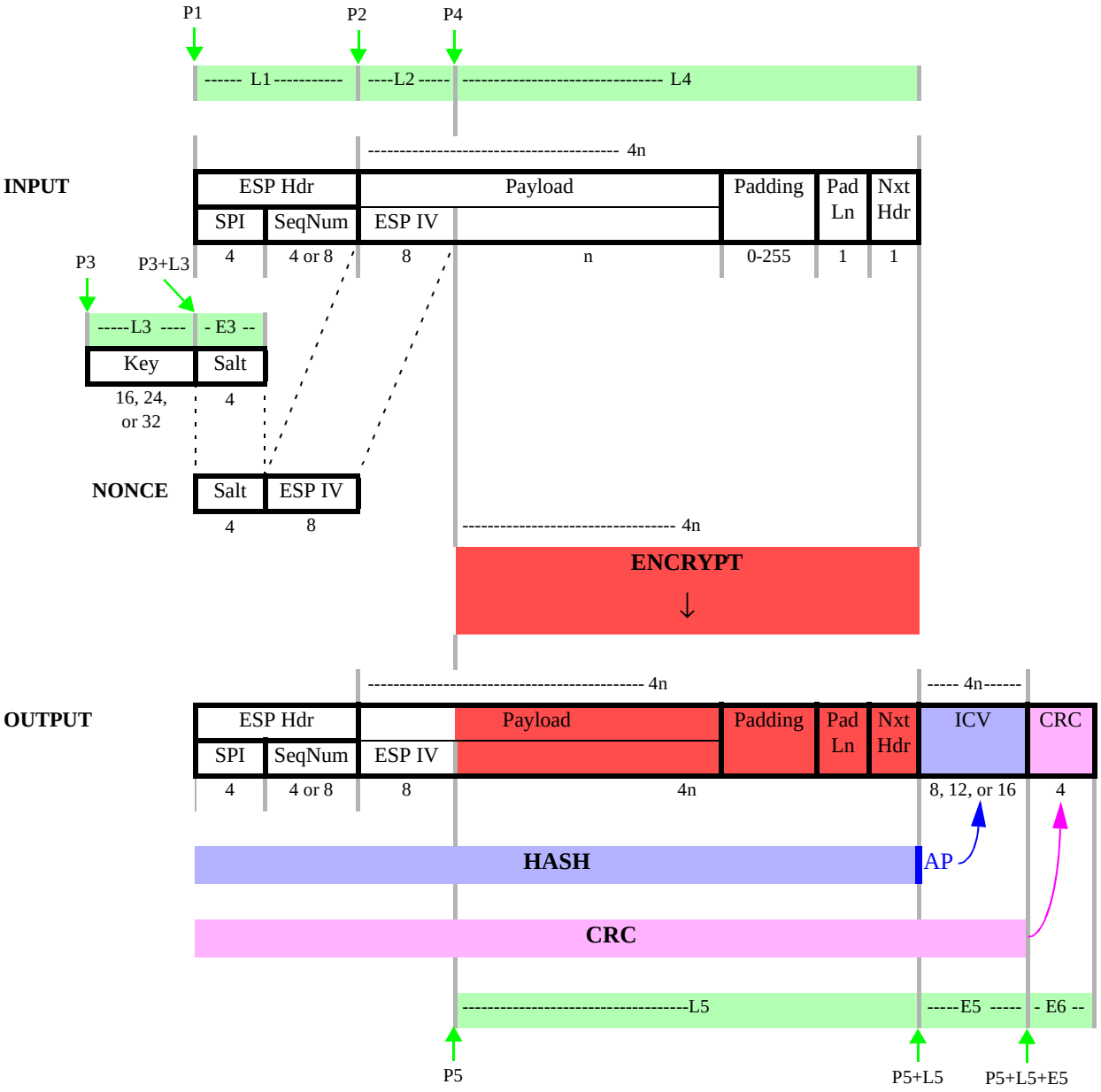


Figure 27-49. AES_GCM Outbound IPsec Packet Pointer Diagram

Notes: **1.** CRC operations are optional.
2. The nonce, AAD, and payload are all sent to the AESU input FIFO. The Nonce and AAD must be padded out the next 16B word, so that the AAD and payload start on a fresh block boundary.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Context In Length	J0	--	-	Eptr0	P0: Address of AES-GCM Context
Pointer 1	GCM-AAD Length	J1	--	-	Eptr1	P1: Address of AES-GCM Additional Authen. Data
Pointer 2	IPsec IV Length	J2	--	-	Eptr2	P2: Pointer to IPsec IV In
Pointer 3	Key Length	J3	Salt Len	-	Eptr3	P3: Address of Cipher Key • IPsec-GCM Salt In
Pointer 4	Plaintext Length	J4	--	-	Eptr4	P4: Address of Plaintext
Pointer 5	Ciphertext Length	J5	ICV Len	-	Eptr5	P5: Address of Ciphertext • ICV Out • CRC Out
Pointer 6	Cipher IV Length	J6	CRC Len	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-50. GCM IPsec Outbound Descriptor Format

27.7.1.2.32 AES_GCM Inbound for IPsec

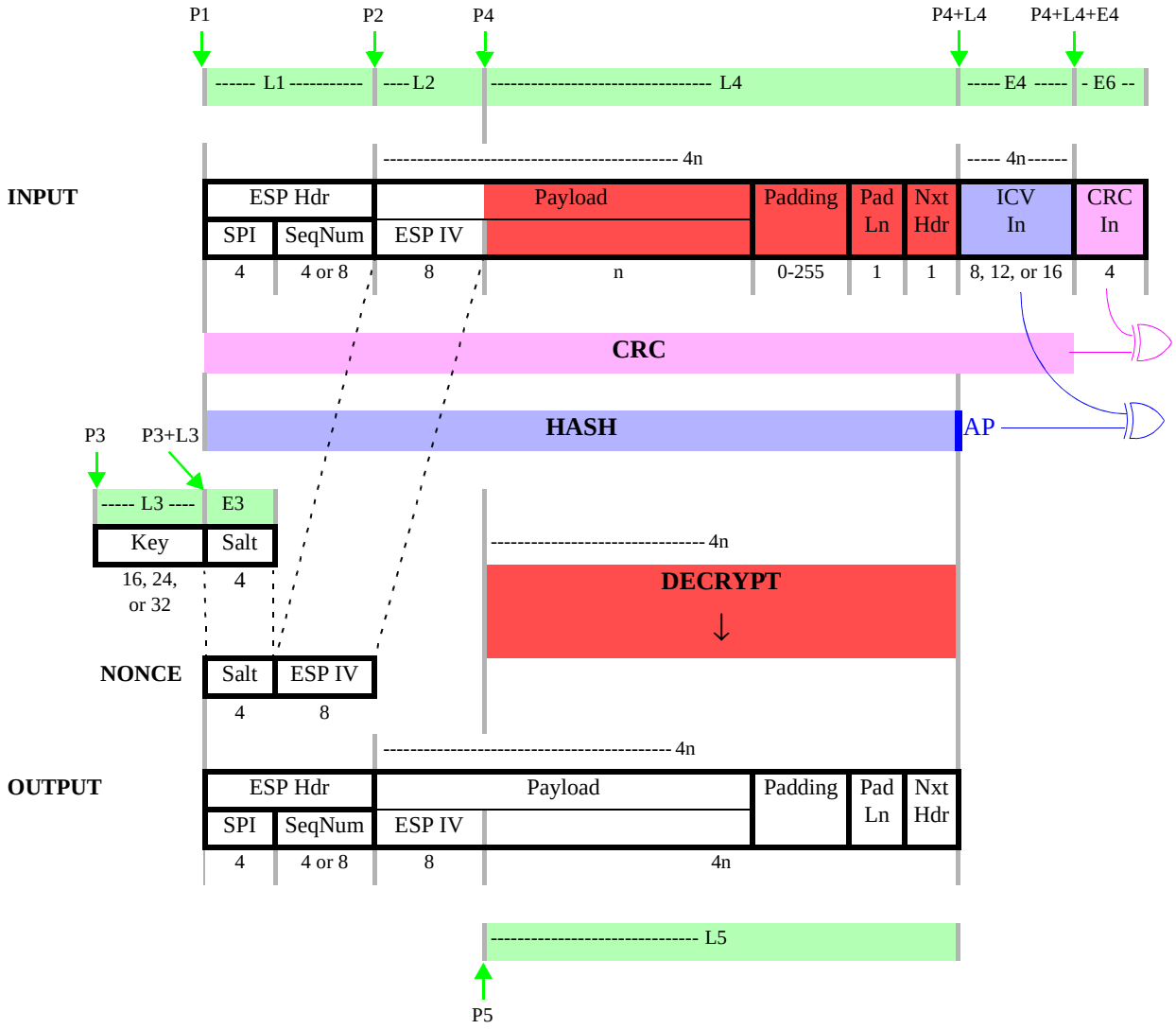


Figure 27-51. AES_GCM Inbound IPsec Packet Pointer Diagram

Notes: **1.** When automatic ICV comparison is used for authentication, “ICV out” is typically not needed, so E5 may be set to 0.

2. The nonce, AAD, and payload are all sent to the AESU input FIFO. The Nonce and AAD must be padded out the next 16B word, so that the AAD and payload start on a fresh block boundary.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Context In Length	J0	--	-	Eptr0	P0: Address of AES-GCM Context
Pointer 1	GCM-AAD Length	J1	--	-	Eptr1	P1: Address of AES-GCM Additional Authen. Data
Pointer 2	IPsec IV Length	J2	--	-	Eptr2	P2: Address of IPsec IV
Pointer 3	Key Length	J3	Salt Len	-	Eptr3	P3: Address of Cipher Key • IPsec-GCM Salt
Pointer 4	Ciphertext Length	J4	ICV Len	-	Eptr4	P4: Address of Ciphertext In • ICV In • CRC In
Pointer 5	Plaintext Length	J5	--	-	Eptr5	P5: Address of Plaintext Out • ICV out • CRC out
Pointer 6	Cipher IV Length	J6	CRC Len	-	Eptr6	P6: Address of Cipher IV Out
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-52. GCM IPsec Inbound Descriptor Format

27.7.1.2.33 Descriptor Type 1100_1: dbl_crc

Table 27-54. Descriptor Format Summary for descriptor dbl_crc

Descriptor Type	field type	Pointer0	Pointer1	Pointer2	Pointer3	Pointer4	Pointer5	Pointer6
1100_1	Length	Header In	Payload In	reserved	reserved	reserved	reserved	reserved
dbl_crc	Extent	Header ICV	Payload ICV	Header ICV Out	Payload ICV Out	reserved	reserved	reserved

27.7.1.2.34 iSCSI dbl_crc Outbound

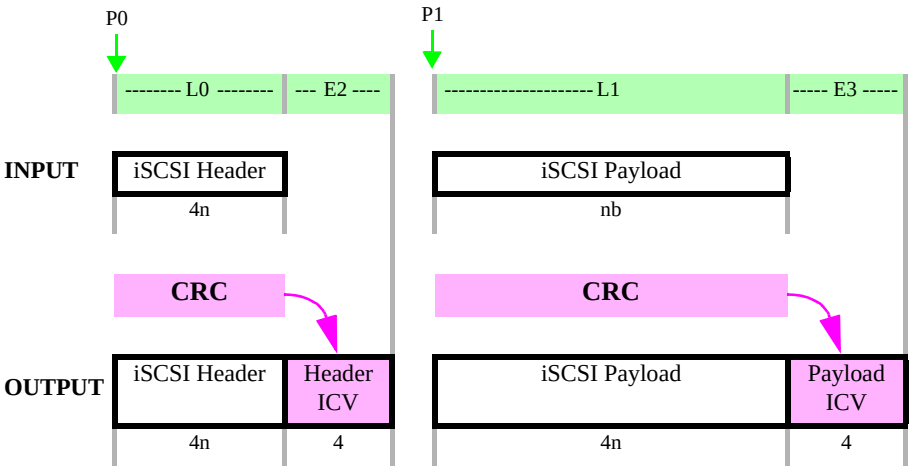


Figure 27-53. SCSI dbl_crc Outbound Packet Pointer Diagram

Notes:

1. The ICV output is placed immediately after the input.
2. The operation shown as “CRC” could alternatively be an MDEU hashing operation.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Header In Length	J0	--	-	Eptr0	P0: Header Data • Header CRC Out
Pointer 1	Payload In Len	J1	--	-	Eptr1	P1: Payload Data • Payload CRC Out
Pointer 2	--	J2	CRC Len	-	Eptr2	P2: --
Pointer 3	--	J3	CRC Len	-	Eptr3	P3: --
Pointer 4	--	J4	--	-	Eptr4	P4: --
Pointer 5	--	J5	--	-	Eptr5	P5: --
Pointer 6	--	J6	--	-	Eptr6	P6: --
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-54. iSCSI dbl_crc Outbound Descriptor Format

Data Processing Steps (as managed by the Channel):

1. Channel sets up CRCU Mode Register and Data Size Register as appropriate.
2. *L0* bytes of header data is fetched from address *P0* and written to CRCU input FIFO.
3. *E2* bytes of computed CRC is fetched from CRCU result register and written to *P0* + *L0*.
4. CRCU is reset.
5. CRCU Mode Register and Data Size Register are re-programmed as appropriate.
6. *L1* bytes of payload data is fetched from address *P1* and written to CRCU input FIFO.
7. *E3* bytes of computed CRC is fetched from CRCU result register and written to *P1*+ *L1*.

Table 27-55. Descriptor Control word examples for dbl_crc Outbound Descriptors

control word (32 bit hex)	Description
800000c8	crc32 IEEE 802 outbound
801000c8	crc32 IETF-3385 outbound

27.7.1.2.35 iSCSI dbl_crc Inbound

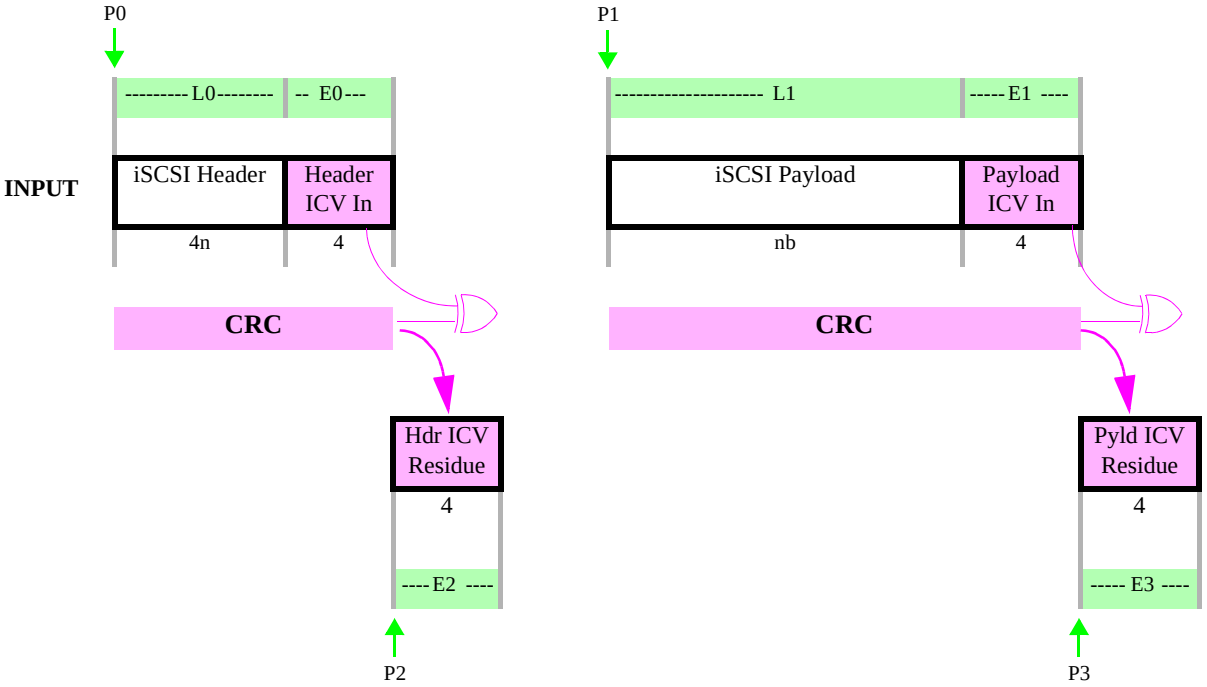


Figure 27-55. iSCSI dbl_crc Inbound Packet Pointer Diagram

Notes:

Notes: **1.** Typically, automatic ICV comparison would be used with this descriptor, so the “ICV out” values (see P2 and P3) are not needed. In this case, P2, E2, P3, and E3 may be set to 0.

2. Failed digest comparisons are indicated by the CICV interrupt from the EU (if this interrupt is enabled), and the ICCR0 and ICCR1 bits in header writeback (if writeback is enabled). ICCR0 gives the result of the payload ICV comparison and ICCR1 gives the result of the header ICV comparison.

	0 15	16	17 23	24 27	28 31	32 63
Header	Descriptor Control					Descriptor Feedback
Pointer 0	Header In Length	J0	CRC Len	-	Eptr0	P0: Address to Header Data • Header CRC In
Pointer 1	Payload In Len	J1	CRC Len		Eptr1	P1: Address to Payload Data • Payload CRC In
Pointer 2	--	J2	CRC Len	-	Eptr2	P2: Address of Header CRC Out
Pointer 3	--	J3	CRC Len	-	Eptr3	P3: Address of Payload CRC Out
Pointer 4	--	J4	--	-	Eptr4	P4: --
Pointer 5	--	J5	--	-	Eptr5	P5: --
Pointer 6	--	J6	--	-	Eptr6	P6: --
	Length	**J**	**Extent**	**--**	**Eptr**	**Pointer**

Figure 27-56. iSCSI dbl_crc Inbound Descriptor Format

Data Processing Steps (as managed by the Channel):

1. Channel sets up CRCU Mode Register and Data Size Register as appropriate.
2. *L0* bytes of header data is fetched from address *P0* and written to CRCU input FIFO.
3. *E0* bytes of received header CRC is fetched from address *P0* + *L0* and written to CRCU input FIFO
4. *E2* bytes of computed CRC is fetched from CRCU result register and written to *P2*.
5. CRCU is reset.
6. CRCU Mode Register and Data Size Register are re-programmed as appropriate.
7. *L1* bytes of payload data is fetched from address *P1* and written to CRCU input FIFO.
8. *E1* bytes of received payload CRC is fetched from address *P1* + *L1* and written to CRCU input FIFO
9. *E3* bytes of computed CRC is fetched from CRCU result register and written to *P3*.

Table 27-56. Descriptor Control word examples for dbl_crc Inbound Descriptors

control word (32 bit hex)	Description
800000ca	crc32 IEEE 802 inbound
840000ca	crc32 IEEE 802 crc check inbound
801000ca	crc32 IETF-3385 inbound
841000ca	crc32 IETF-3385 crc check inbound

27.7.2 Pointers

Bit	63 48	47	46 40	39 36	35 32	31 0
	LENGTH	J	EXTENT	—	EPTR	POINTER

The descriptor contains seven pointers that define where the SEC accesses its input and output data parcels in memory. The pointers are numbered from 0 to 6 as shown in **Section 27.7.1.1**. The channel determines how it uses each of the pointers based on the Descriptor Type and Direction fields in the header. The channel accesses the first data parcel by starting at a location given by a POINTER value, and accessing a number of bytes given by a LENGTH or EXTENT value. Subsequent data parcels can be accessed by starting where a previous data parcel ended, or by starting at a different POINTER location. The LENGTH or EXTENT used with any POINTER can be from the same pointer or from a different pointer in the same descriptor. Although the

EXTENT field exists in each pointer of the SEC descriptor, only the EXTENTs in pointers 3, 4, and 5 are currently used. If Extend Address Enable is set (1), then the four EPTR bits are concatenated with the POINTER field to form a 36-bit pointer address.

Table 27-57. Pointer Field Definitions

Bits	Name	Description	Settings
63–48	LENGTH	**Length** The use of this field depends on the Descriptor Type and Direction in the header. A value of zero causes the channel to skip this pointer.	A number of bytes in the range 0 to 65535.
47	J	**Jump** Determines whether to jump to a link table whenever the POINTER field in this same field is used.	0 The POINTER field points to data. 1 The POINTER field points to a link table, and scatter/gather is enabled.
46–40	EXTENT	**Extent** A number of bytes in the range 0 to 127. The use of this field depends on the Descriptor Type and Direction in the header.	
39–36	—	Reserved	
35–32	EPTR	**Extended Pointer** Concatenated as the top 4 bits of the pointer when EAE is high (see the EAE bit in **Table 27-1**).	
31–0	POINTER	**Pointer** A memory address.	

Examples of special cases include:

- Sometimes, a descriptor field does not apply for the requested service. With seven pointers, it is possible that not all these pointers are required to specify the input and output parameters. (Some operations, for example, do not require context.) Where a particular pointer is not used, all fields in that pointer should be cleared (0).
- Some descriptors involve more than seven parcels of input and output data. In these cases, it is necessary to use one POINTER field to address a sequence of data parcels.
- LENGTH and EXTENT fields normally specify the sizes of data parcels. Often, but not always, the size of a parcel located at the address contained in the matching POINTER field. Sometimes an EXTENT field will refer to data in a pointer not in the same pointer. For example, with the CCMP descriptor type, the length of the CRC check field appears in E0, but the field that is E0 bytes in length is referred to either by P4 or P5, depending on the direction bit in the Header Control Word. In some cases, however, the POINTER field is zero, and the LENGTH and/or EXTENT fields simply specify values to write to an EU.

- The J bit in each pointer is used to enable the scatter/gather feature. If a data parcel to be read or written by SEC is in one contiguous block of memory locations, then the scatter/gather feature is not needed. In this case, set the POINTER to point directly at the first byte of the parcel, and the J bit should be 0. On the other hand, if the data parcel is stored in several separate segments of memory, then the scatter/gather capability is needed to assemble or distribute the complete parcel. In this case, set the POINTER to point to a link table, and the J bit should be 1. For link table format, see **Section 27.7.3**, *Link Tables*, on page 27-175. Scatter/gather capability is available for all pointers of all descriptor types, with the exception that the raid-xor descriptor type does not allow scatter/gather.

27.7.3 Link Tables

Bit	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	SEGLEN																—						R	N	—				EPTR			

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
Field	SEGPTR																															

Link tables implement scatter/gather capability. For gather operations, a link table specifies a list of memory segments that are to be concatenated in the process of assembling parcels. For scatter operations, a link table specifies a list of memory segments into which the output data should be written. Scatter or gather of a parcel may be specified by a single link table or by a chain of link tables that are linked together with pointers.

The link table or chain of link tables accessed through some descriptor POINTER must specify enough memory segments to hold precisely all the data that will be accessed through that pointer. In most cases, only a single parcel is accessed through a given POINTER, and the chain of link tables specifies just that parcel. In other cases, the descriptor POINTER is used multiple times to access a sequence of parcels, and the chain of link tables must supply data for the entire sequence.

A link table can contain any number of 64-bit entries. There are two kinds of entries, regular entries and next entries. Each regular entry specifies a memory segment by means of a 36-bit starting address (SEGPTR) and a 16-bit length (SEGLEN). A next entry is used at the end of a link table to specify that the list of memory segments is continued in another link table. In a next entry, the N bit is set, the SEGPTR field gives the address of the next link table, and the SEGLEN field must be 0. A chain of link tables can contain any number of link tables.

Whether the list of memory segments is in a single link table or split into several link tables, the last entry in the last link table is a regular entry with the R (return) bit set. The R bit signifies the end of link table operations so that the channel returns to the descriptor for its next pointer (if any).

Table 27-58 the components in a link table field.

Table 27-58. Link Table Field Definitions

Fields	Description	Settings
SEGLEN 63–48	**Segment Length** Indicates the number of bytes in the memory segment. **Notes:** **1.** See Section 27.7.6.2, **Channel Status Registers (CSR[1–4])**). **2.** When N = 1, must be 0.	N = 0 0 Error state in G_STATE (gather) or S_STATE (scatter). Any other value (1–65535) indicates the number of bytes in the memory segment. N = 1 0 Required value 1 Not valid.
— 47–42	Reserved	
R 41	**Return** Specifies whether this is the last data parcel. **Note:** If this entry does not specify the correct number of bytes to complete the last data parcel, a G_STATE or S_STATE error is set in the Channel Pointer Register (see Section 27.7.6.2).	N = 0 0 No special action. 1 Last entry in the chain of link tables. N = 1 All values ignored.
N 40	**Next** Indicates whether this is the last link table entry. The SEGPTR field holds the address of the next link table in the chain.	0 No special action. 1 Last 8-byte entry in the current link table.
— 39–36	Reserved	
EPTR 35–32	**Extended Pointer** Concatenated as the top 4 bits of the segment pointer when EAE is set (see the EAE bit in **Table 27-1**).	
SEGPTR 31–0	**Segment Pointer** A memory address.	

27.7.4 SEC Controller

The following sections describe the registers and structures used by the SEC Controller.

27.7.4.1 Master Control Register (MCR)

MCR — Master Control Register — Offset 0xC1030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	RCA1	RCA2	RCA3	RCA4	—		PRIORITY		—						GIH	SWR
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CHN3_EU_PR_CNT								CHN4_EU_PR_CNT							
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CHN3_BUS_PR_CNT								CHN4_BUS_PR_CNT							
Type	R/W															
Reset	0x0000															

The MCR controls certain functions in the controller and provides a means for software to reset the SEC. **Table 27-59** describes the Master Control Register bit fields.

Table 27-59. Master Control Register Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–48	0	Reserved. Write to zero for future compatibility.	
RCA1 47	0	**Remap Channel Address 1** Writing 1 to this bit remaps all Channel 1 core-accessible registers to an alternate 4KB address range. The channel done and error bits in the Interrupt Status Register cause the secondary interrupt to assert instead of the primary interrupt.	0 Channel 1 registers are accessible in the 0x11xx address range (default) 1 Channel 1 registers are accessible in the 0x01xx address range
RCA2 46	0	**Remap Channel Address 2** Writing 1 to this bit remaps all Channel 2 core-accessible registers to an alternate 4KB address range. The channel done and error bits in the Interrupt Status Register cause the secondary interrupt to assert instead of the primary interrupt.	0 Channel 2 registers are accessible in the 0x12xx address range (default) 1 Channel 2 registers are accessible in the 0x02xx address range

Table 27-59. Master Control Register Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
RCA3 45	0	**Remap Channel Address 3** Writing 1 to this bit remaps all Channel 3 core-accessible registers to an alternate 4KB address range. The channel done and error bits in the Interrupt Status Register cause the secondary interrupt to assert instead of the primary interrupt.	0 Channel 3 registers are accessible in the 0x13xx address range (default) 1 Channel 3 registers are accessible in the 0x03xx address range
RCA4 44	0	**Remap Channel Address 4** Writing 1 to this bit remaps all Channel 4 core-accessible registers to an alternate 4KB address range. The channel done and error bits in the Interrupt Status Register cause the secondary interrupt to assert instead of the primary interrupt.	0 Channel 4 registers are accessible in the 0x14xx address range (default) 1 Channel 4 registers are accessible in the 0x04xx address range
— 43–42	0	Reserved. Write to zero for future compatibility.	
PRIORITY 41–40	0	**Priority on Master Bus** The setting of these bits determines the transaction priority level the SEC asserts to the CLASS internal arbiter. The SEC does not dynamically alter its priority level based on system congestion or SEC utilization, however software can change the SEC priority level in realtime.	00 Lowest Priority (default) 01 Next Lowest Priority 10 Next Highest Priority 11 Highest Priority
— 39–34	0	Reserved. Write to zero for future compatibility.	
GIH 33	0	**Global Inhibit** Writing 1 to this bit defines external bus transfers to the master bus as non-snoopable and results in reducing the number of snoop bus requests generated by the external bridge.	0 External transfers snoopable (default) 1 External transfers are not snoopable.
SWR 32	0	**Software Reset** Writing 1 to this bit causes a global software reset. Upon completion of the reset, this bit is automatically cleared.	0 Do not reset 1 Global Reset
CHN3_EU_PR_CNT 31–24	0	**Channel 3 EU Priority Counter** This counter is used by the controller to determine when Channel 3 is denied access to a requested EU long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero and the CHN4_EU_PR_CTR is also set to zero, the controller assigns EUs on a purely round-robin basis. If either counter is zero and the other is non-zero, the zero is interpreted as 256.	
CHN4_EU_PR_CNT 23–17	0	**Channel 4 EU Priority Counter** This counter is used by the controller to determine when Channel 4 is denied access to a requested EU long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero and the CHN3_EU_PR_CTR is also set to zero, the controller assigns EUs on a purely round-robin basis. If either counter is zero and the other is non-zero, the zero is interpreted as 256.	

Table 27-59. Master Control Register Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
CHN3_BUS_PR_CNT 16–8	0	**Channel 3 Bus Priority Counter** This counter is used by the controller to determine when Channel 3 is denied access to the polychannel long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero and the CHN4_BUS_PR_CTR is also set to zero, the controller assigns polychannel access on a purely round-robin basis. If either counter is zero and the other is non-zero, the zero is interpreted as 256.	
CHN4_BUS_PR_CNT 7–0	0	**Channel 4 Bus Priority Counter** This counter is used by the controller to determine when Channel 4 is denied access to the polychannel long enough to warrant immediate elevation to level 2 priority. **Note:** If set to zero and the CHN3_BCUS_PR_CTR is also set to zero, the controller assigns polychannel access on a purely round-robin basis. If either counter is zero and the other is non-zero, the zero is interpreted as 256.	

27.7.4.2 Controller Identification Register (CIDR)

CIDR — Controller Identification Register — Offset 0xC1020

Bits	63 … 0
	ID
Type	R
Reset	0x0030030100000000

The read-only ID register contains the same value as the IP Block Revision Register (see **Section 27.7.4.3**, *Controller IP Block Revision Register (CIPBRR)*). In updating existing software, use of this address is likely to be most convenient, since it is the same address used for version information in previous SEC revisions.

27.7.4.3 Controller IP Block Revision Register (CIPBRR)

CIPBRR — Controller IP Block Revision Register — Offset 0xC1BF8

Bits	63–48	47–40	39–32	31–24	23–16	15–8	7–0
Field	IP_ID	IP_MJ	IP_MN	—	IP_INT	—	IP_CFG
Type	R						
Reset	0x0030	0x03	0x01	0x00	0x00	0x00	0x00

The read-only Controller IP Block Revision Register contains a 64-bit value that uniquely identifies the version of the SEC 3.1 block. **Table 27-60** describes the fields of the IP Block Revision Register.

Table 27-60. CIPBRR Bit Field Descriptions

Bits	Reset	Description	Settings
IP_ID 63–48	0x0030	**IP Block Identifier** Specifies the revision level of this IP block.	0x0030 fixed value.
IP_MJ 47–40	0x03	**IP Major Revision Number** Major revision level value.	0x03 fixed value.
IP_MN 39–32	0x01	**IP Minor Revision Number** Minor revision level value.	0x01 fixed value.
— 31–24	0	Reserved. Write to zero for future compatibility.	
IP_INT 23–16	0	**IP Integration Options**	0x00 fixed value.
— 15–8	0	Reserved. Write to zero for future compatibility.	
IP_CFG 7–0	0	**IP Block Configuration Options**	0x00 fixed value.

27.7.4.4 EU Assignment Status (EUASR)

EUASR — EU Assignment Status Register — Offset 0xC1028

Bits	63–60	59–56	55–52	51–48	47–44	43–40	39–36	35–32
Field	—	AFEU	—	MDEU	—	AESU	—	DEU
Type	R							
Reset	0xF	0x0	0xF	0x0	0xF	0x0	0xF	0x0

Bits	31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
Field	—	—	CRCU	KEU	STEU	PKEU	—	RNGU
Type	R							
Reset	0xFF		0x0	0x0	0x0	0x0	0xF	0x0

The EUASR is used to check the assignment status of an EU to a particular channel. When an EU is already assigned, it is inaccessible to any other channel. **Table 27-61** describes the fields of the EUASR.

Table 27-61. EUASR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–60	0xF	Reserved. Write to 0xF for future compatibility.	
AFEU 59–56	0x0	**AFEU** Indicates the AFEU channel assignment.	See **Table 27-62**.
— 55–52	0xF	Reserved. Write to 0xF for future compatibility.	
MDEU 51–48	0x0	**MDEU** Indicates the MDEU channel assignment.	See **Table 27-62**.
— 47–44	0xF	Reserved. Write to 0xF for future compatibility.	
AESU 43–40	0x0	**AESU** Indicates the AESU channel assignment.	See **Table 27-62**.
— 39–36	0xF	Reserved. Write to 0xF for future compatibility.	
DEU 35–32	0x0	**DEU** Indicates the DEU channel assignment.	See **Table 27-62**.
— 31–28	0xF	Reserved. Write to zero for future compatibility.	
— 27–24	0xF	Reserved. Write to zero for future compatibility.	
CRCU 23–20	0x0	**CRCU** Indicates the CRCU channel assignment.	See **Table 27-62**.
KEU 19–16	0x0	**KEU** Indicates the KEU channel assignment.	See **Table 27-62**.
STEU 15–12	0x0	**STEU** Indicates the STEU channel assignment.	See **Table 27-62**.
PKEU 11–8	0x00	**PKEU** Indicates the PKEU channel assignment.	See **Table 27-62**.
— 7–4	0	Reserved. Write to 0xF for future compatibility.	
RNGU 3–0	0x00	**RNGU** Indicates the RNGU channel assignment.	See **Table 27-62**.

Table 27-62. Channel Assignment Value

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3
0x4	Channel 4
0x5–0xE	Undefined
0xF	Unavailable

27.7.4.5 Controller Interrupt Enable Register (CIER)

CIER — Controller Interrupt Enable Register — Offset 0xC1008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—								FFE_CNT	DF_CNT	DI_CNT	DO_CNT	—			ITO
Subfield	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—				DONE Overflow				CHN_4		CHN_3		CHN_2		CHN_1	
Subfield					CH4	CH3	CH2	CH1	Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—				CRCU		KEU		STEU		PKEU		—		RNGU	
Subfield					Err	Dn	Err	Dn	Err	Dn	Err	Dn			Err	Dn
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		AFEU		—		MDEU		—		AESU		—		DEU	
Subfield			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Type	R/W															
Reset	0x0000															

The SEC controller generates a single interrupt output from all possible interrupt sources. These sources can be individually enabled by the Interrupt Enable Register. If enabled, the interrupt source value, when active, is captured into the Interrupt Status Register. Each interrupt source is individually enabled by setting its corresponding bit. At reset, all bits are disabled. For normal operation, leave channel interrupts enabled, while disabling interrupts from the EUs. The channels generate the appropriate interrupts to the core processor. **Table 27-63** describes the register field names in the CIER.

Note: The recommended value for this register is 0x00310FFF00000000. Execution Unit interrupt bits are provided as a convenience during debug.

Table 27-63. CIER Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–56	0	Reserved. Write to zero for future compatibility.	
FFE_CNT 55	0	**Fetch FIFO Enqueue Count Rollover** Indicates whether the Fetch FIFO enqueue counter has rollover detect is enabled.	0 Timer rollover detect not enabled. 1 Timer rollover detect enabled.
DF_CNT 54	0	**Descriptor Finished Count Rollover** Indicates whether the Descriptor Finished Counter rollover detect is enabled.	0 Counter rollover detect not enabled. 1 Counter rollover detect enabled.
DI_CNT 53	0	**Data In Count Rollover** Indicates whether the Data In Counter rollover detect is enabled.	0 Counter rollover detect not enabled. 1 Counter rollover detect enabled.
DO_CNT 52	0	**Data Out Count Rollover** Indicates whether the Data Out Counter rollover detect is enabled.	0 Counter rollover detect not enabled. 1 Counter rollover detect enabled.
— 51–49	0	Reserved. Write to zero for future compatibility.	
ITO 48	0	**Internal Time Out** Indicates whether internal time-out detection is enabled.	0 Internal time-out interrupt disabled. 1 Internal time-out interrupt enabled.
— 47–44	0	Reserved. Write to zero for future compatibility.	
DONE Overflow CH4–CH1 43–40	0	**DONE Overflow for Channels 4–1** For each channel, the bit enables/disables the DONE overflow detection.	0 DONE overflow interrupt disabled. 1 DONE overflow interrupt enabled.
CHN4 Err 39	0	**Channel 4 Error Interrupt** Indicates whether a Channel 4 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 Error detection interrupt disabled. 1 Error detection interrupt enabled.
CHN4 Dn 38	0	**Channel 4 Done Interrupt** Indicates whether Channel 4 has completed its operation.	0 Done detection interrupt disabled. 1 Done detection interrupt enabled.
CHN3 Err 37	0	**Channel 3 Error Interrupt** Indicates whether a Channel 3 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 Error detection interrupt disabled. 1 Error detection interrupt enabled.
CHN3 Dn 36	0	**Channel 3 Done Interrupt** Indicates whether Channel 3 has completed its operation.	0 Done detection interrupt disabled. 1 Done detection interrupt enabled.
CHN2 Err 35	0	**Channel 2 Error Interrupt** Indicates whether a Channel 2 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 Error detection interrupt disabled. 1 Error detection interrupt enabled.

Table 27-63. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
CHN2 Dn 34	0	**Channel 2 Done Interrupt** Indicates whether Channel 2 has completed its operation.	0 Done detection interrupt disabled. 1 Done detection interrupt enabled.
CHN1 Err 33	0	**Channel 1 Error Interrupt** Indicates whether a Channel 1 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 Error detection interrupt disabled. 1 Error detection interrupt enabled.
CHN1 Dn 32	0	**Channel 1 Done Interrupt** Indicates whether Channel 1 has completed its operation.	0 Done detection interrupt disabled. 1 Done detection interrupt enabled.
— 31–28	0	Reserved. Write to zero for future compatibility.	
CRCU Err 25	0	**CRCU Error Interrupt** Indicates whether the CRCU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
CRCU Dn 24	0	**CRCU Done** Indicates whether the CRCU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
KEU Err 25	0	**KEU Error Interrupt** Indicates whether the KEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
KEU Dn 24	0	**KEU Done** Indicates whether the KEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
STEU Err 23	0	**STEU Error Interrupt** Indicates whether the STEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
STEU Dn 22	0	**STEU Done** Indicates whether the STEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
PKEU Err 21	0	**PKEU Error Interrupt** Indicates whether the PKEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
PKEU Dn 20	0	**PKEU Done** Indicates whether the PKEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
— 19–18	0	Reserved. Write to zero for future compatibility.	

Table 27-63. CIER Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
RNGU Err 17	0	**RNGU Error Interrupt** Indicates whether the RNGU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
RNGU Dn 16	0	**RNGU Done** Indicates whether the RNGU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
— 15–14	0	Reserved. Write to zero for future compatibility.	
AFEU Err 13	0	**AFEU Error Interrupt** Indicates whether the AFEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
AFEU Dn 12	0	**AFEU Done** Indicates whether the AFEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
— 11–10	0	Reserved. Write to zero for future compatibility.	
MDEU Err 9	0	**MDEU Error Interrupt** Indicates whether the MDEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
MDEU Dn 8	0	**MDEU Done** Indicates whether the MDEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
— 7–6	0	Reserved. Write to zero for future compatibility.	
AESU Err 5	0	**AESU Error Interrupt** Indicates whether the AESU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
AESU Dn 4	0	**AESU Done** Indicates whether the AESU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.
— 3–2	0	Reserved. Write to zero for future compatibility.	
DEU Err 1	0	**DEU Error Interrupt** Indicates whether the DEU generated an error.	0 Error interrupt disabled. 1 Error interrupt enabled.
DEU Dn 0	0	**DEU Done** Indicates whether the DEU has completed its operation.	0 Done interrupt disabled. 1 Done interrupt enabled.

27.7.4.6 Controller Interrupt Status Register (CISR)

CISR — Controller Interrupt Status Register — Offset 0xC1010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—								FFE_CNT	DF_CNT	DI_CNT	DO_CNT	—			ITO
Subfield	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—				DONE Overflow				CHN_4		CHN_3		CHN_2		CHN_1	
Subfield					CH4	CH3	CH2	CH1	Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000															
R/W	R															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—				CRCU		KEU		STEU		PKEU		—		RNGU	
Subfield					Err	Dn	Err	Dn	Err	Dn	Err	Dn			Err	Dn
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		AFEU		—		MDEU		—		AESU		—		DEU	
Subfield			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Type	R															
Reset	0x0000															

The CISR contains fields representing all possible sources of interrupts. The Interrupt Status Register is cleared either by a reset, or by writing the appropriate bits active in the CICR (see **Section 27.7.4.7**, *Controller Interrupt Clear Register (CICR)*, on page 27-189. The CISR bit fields are described in **Table 27-64**.

Table 27-64. CISR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–56	0	Reserved. Write to zero for future compatibility.	
FFE_CNT 55	0	**Fetch FIFO Enqueue Count Rollover** Indicates whether the Fetch FIFO enqueue counter has rolled over to zero.	0 Timer not rolled over. 1 Timer rolled over.
DF_CNT 54	0	**Descriptor Finished Count Rollover** Indicates whether the Descriptor Finished Counter rolled over to zero.	0 Counter not rolled over. 1 Counter rolled over.
DI_CNT 53	0	**Data In Count Rollover** Indicates whether the Data In Counter rolled over to zero.	0 Counter not rolled over. 1 Counter rolled over.
DO_CNT 52	0	**Data Out Count Rollover** Indicates whether the Data Out Counter rolled over to zero.	0 Counter not rolled over. 1 Counter rolled over.
— 51–49	0	Reserved. Write to zero for future compatibility.	

Table 27-64. CISR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
ITO 48	0	**Internal Time Out** Indicates whether an internal time-out has occurred, that is, whether a channel or EU has failed to respond to a slave read or write within 16 cycles, which only occurs in an impending hang condition. Assertion of this interrupt indicates the SEC Controller has completed the transaction, but the completed transaction does not result in a successful read or write. The interrupt advises the system that the slave transaction was unsuccessful.	0 No internal time-out. 1 An internal time-out was detected.
— 47–44	0	Reserved. Write to zero for future compatibility.	
DONE Overflow CH4–CH1 43–40	0	**DONE Overflow for Channels 4–1** For each channel, the bit indicates whether a DONE overflow condition occurred. A DONE overflow occurs when more than 15 Done interrupts are queued from the associated channel without an interrupt clear from the core processor.	0 No DONE overflow. 1 DONE overflow error.
CHN4 Err 39	0	**Channel 4 Error Interrupt** Indicates whether a Channel 4 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 No error detected. 1 Error detected.
CHN4 Dn 38	0	**Channel 4 Done Interrupt** Indicates whether Channel 4 has completed its operation.	0 Not done. 1 Operation done.
CHN3 Err 37	0	**Channel 3 Error Interrupt** Indicates whether a Channel 3 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 No error detected. 1 Error detected.
CHN3 Dn 36	0	**Channel 3 Done Interrupt** Indicates whether Channel 3 has completed its operation.	0 Not done. 1 Operation done.
CHN2 Err 35	0	**Channel 2 Error Interrupt** Indicates whether a Channel 2 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 No error detected. 1 Error detected.
CHN2 Dn 34	0	**Channel 2 Done Interrupt** Indicates whether Channel 2 has completed its operation.	0 Not done. 1 Operation done.
CHN1 Err 33	0	**Channel 1 Error Interrupt** Indicates whether a Channel 1 error occurred. The Channel Status Register must be read to determine the exact cause of the error.	0 No error detected. 1 Error detected.
CHN1 Dn 32	0	**Channel 1 Done Interrupt** Indicates whether Channel 1 has completed its operation.	0 Not done. 1 Operation done.
— 31–28	0	Reserved. Write to zero for future compatibility.	
CRCU Err 27	0	**CRCU Error Interrupt** Indicates whether the CRCU generated an error.	0 No error detected. 1 Error detected.
CRCU Dn 26	0	**CRCU Done** Indicates whether the CRCU has completed its operation.	0 Not done. 1 Operation done.
KEU Err 25	0	**KEU Error Interrupt** Indicates whether the KEU generated an error.	0 No error detected. 1 Error detected.
KEU Dn 24	0	**KEU Done** Indicates whether the KEU has completed its operation.	0 Not done. 1 Operation done.
STEU Err 23	0	**STEU Error Interrupt** Indicates whether the STEU generated an error.	0 No error detected. 1 Error detected.

Table 27-64. CISR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
STEU Dn 22	0	**STEU Done** Indicates whether the STEU has completed its operation.	0 Not done. 1 Operation done.
PKEU Err 21	0	**PKEU Error Interrupt** Indicates whether the PKEU generated an error.	0 No error detected. 1 Error detected.
PKEU Dn 20	0	**PKEU Done** Indicates whether the PKEU has completed its operation.	0 Not done. 1 Operation done.
— 19–18	0	Reserved. Write to zero for future compatibility.	
RNGU Err 17	0	**RNGU Error Interrupt** Indicates whether the RNGU generated an error.	0 No error detected. 1 Error detected.
RNGU Dn 16	0	**RNGU Done** Indicates whether the RNGU has completed its operation.	0 Not done. 1 Operation done.
— 15–14	0	Reserved. Write to zero for future compatibility.	
AFEU Err 13	0	**AFEU Error Interrupt** Indicates whether the AFEU generated an error.	0 No error detected. 1 Error detected.
AFEU Dn 12	0	**AFEU Done** Indicates whether the AFEU has completed its operation.	0 Not done. 1 Operation done.
— 11–10	0	Reserved. Write to zero for future compatibility.	
MDEU Err 9	0	**MDEU Error Interrupt** Indicates whether the MDEU generated an error.	0 No error detected. 1 Error detected.
MDEU Dn 8	0	**MDEU Done** Indicates whether the MDEU has completed its operation.	0 Not done. 1 Operation done.
— 7–6	0	Reserved. Write to zero for future compatibility.	
AESU Err 5	0	**AESU Error Interrupt** Indicates whether the AESU generated an error.	0 No error detected. 1 Error detected.
AESU Dn 4	0	**AESU Done** Indicates whether the AESU has completed its operation.	0 Not done. 1 Operation done.
— 3–2	0	Reserved. Write to zero for future compatibility.	
DEU Err 1	0	**DEU Error Interrupt** Indicates whether the DEU generated an error.	0 No error detected. 1 Error detected.
DEU Dn 0	0	**DEU Done** Indicates whether the DEU has completed its operation.	0 Not done. 1 Operation done.

27.7.4.7 Controller Interrupt Clear Register (CICR)

CICR — Controller Interrupt Clear Register — Offset 0xC1018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—								FFE_CNT	DF_CNT	DI_CNT	DO_CNT	—			ITO
Subfield	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—				DONE Overflow				CHN_4		CHN_3		CHN_2		CHN_1	
Subfield					CH4	CH3	CH2	CH1	Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—				CRCU		KEU		STEU		PKEU		—		RNGU	
Subfield					Err	Dn	Err	Dn	Err	Dn	Err	Dn			Err	Dn
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		AFEU		—		MDEU		—		AESU		—		DEU	
Subfield			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Type	R/W															
Reset	0x0000															

The CICR provides a means of clearing the CISR. When a 1 is written to a bit in the CICR, the corresponding bit in the CISR is cleared, clearing the interrupt output pin $\overline{\text{IRQ}}$ (assuming the cleared bit in the CISR is the only interrupt source). If the input source to the CISR is a steady-state signal that remains active, the appropriate CISR bit, and subsequently $\overline{\text{IRQ}}$, is reasserted shortly thereafter. The bit fields are described in **Table 27-65**.

Note: When an ICR bit is written, it automatically clears itself one cycle later. It is not necessary to write a 0 to a bit position to which a 1 is written.

Interrupts are registered and sent based upon the conditions that cause them. If the cause of an interrupt is not removed, the interrupt returns a few cycles after it is cleared using the CICR.

Table 27-65. CICR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–56	0	Reserved. Write to zero for future compatibility.	
FFE_CNT 55	0	**Fetch FIFO Enqueue Count Rollover** Used to clear the fetch FIFO enqueue counter rollover status bit.	0 No action. 1 Clear status bit.
DF_CNT 54	0	**Descriptor Finished Count Rollover** Used to clear the descriptor finished counter rollover status bit.	0 No action. 1 Clear status bit.

Table 27-65. CICR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
DI_CNT 53	0	**Data In Count Rollover** Used to clear the data in counter rollover error status bit.	0 No action. 1 Clear status bit.
DO_CNT 52	0	**Data Out Count Rollover** Used to clear the data out counter rollover error status bit.	0 No action. 1 Clear status bit.
— 51–49	0	Reserved. Write to zero for future compatibility.	
ITO 48	0	**Internal Time Out** Used to clear the time-out error status bit.	0 No action. 1 Clear status bit.
— 47–44	0	Reserved. Write to zero for future compatibility.	
DONE Overflow CH4–CH1 43–40	0	**DONE Overflow for Channels 4–1** Used to clear the DONE overflow error status bit for each channel.	0 No action. 1 Clear status bit.
CHN4 Err 39	0	**Channel 4 Error Interrupt** Clears the Channel 4 error status bit	0 No action. 1 Clear status bit.
CHN4 Dn 38	0	**Channel 4 Done Interrupt** Clears the Channel 4 done status bit.	0 No action. 1 Clear status bit.
CHN3 Err 37	0	**Channel 3 Error Interrupt** Clears the Channel 3 error status bit	0 No action. 1 Clear status bit.
CHN3 Dn 36	0	**Channel 3 Done Interrupt** Clears the Channel 3 done status bit.	0 No action. 1 Clear status bit.
CHN2 Err 35	0	**Channel 2 Error Interrupt** Clears the Channel 2 error status bit.	0 No action. 1 Clear status bit.
CHN2 Dn 34	0	**Channel 2 Done Interrupt** Clears the Channel 2 done status bit	0 No action. 1 Clear status bit.
CHN1 Err 33	0	**Channel 1 Error Interrupt** Clears the Channel 1 error status bit.	0 No action. 1 Clear status bit.
CHN1 Dn 32	0	**Channel 1 Done Interrupt** Clears the Channel 1 done status bit.	0 No action. 1 Clear status bit.
— 31–28	0	Reserved. Write to zero for future compatibility.	
CRCU Err 27	0	**CRCU Error Interrupt** Clears the CRCU error status bit.	0 No action. 1 Clear status bit.
CRCU Dn 26	0	**CRCU Done** Clears the CRCU done status bit.	0 No action. 1 Clear status bit.
KEU Err 25	0	**KEU Error Interrupt** Clears the KEU error status bit.	0 No action. 1 Clear status bit.
KEU Dn 24	0	**KEU Done** Clears the KEU done status bit.	0 No action. 1 Clear status bit.
STEU Err 23	0	**STEU Error Interrupt** Clears the STEU error status bit.	0 No action. 1 Clear status bit.
STEU Dn 22	0	**STEU Done** Clears the STEU done status bit.	0 No action. 1 Clear status bit.
PKEU Err 21	0	**PKEU Error Interrupt** Clears the PKEU error status bit.	0 No action. 1 Clear status bit.
PKEU Dn 20	0	**PKEU Done** Clears the PKEU done status bit.	0 No action. 1 Clear status bit.

Table 27-65. CICR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
— 19–18	0	Reserved. Write to zero for future compatibility.	
RNGU Err 17	0	**RNGU Error Interrupt** Clears the RNGU error status bit.	0 No action. 1 Clear status bit.
RNGU Dn 16	0	**RNGU Done** Clears the RNGU done status bit.	0 No action. 1 Clear status bit.
— 15–14	0	Reserved. Write to zero for future compatibility.	
AFEU Err 13	0	**AFEU Error Interrupt** Clears the AFEU error status bit.	0 No action. 1 Clear status bit.
AFEU Dn 12	0	**AFEU Done** Clears the AFEU done status bit.	0 No action. 1 Clear status bit.
— 11–10	0	Reserved. Write to zero for future compatibility.	
MDEU Err 9	0	**MDEU Error Interrupt** Clears the MDEU error status bit.	0 No action. 1 Clear status bit.
MDEU Dn 8	0	**MDEU Done** Clears the MDEU done status bit.	0 No action. 1 Clear status bit.
— 7–6	0	Reserved. Write to zero for future compatibility.	
AESU Err 5	0	**AESU Error Interrupt** Clears the AESU error status bit.	0 No action. 1 Clear status bit.
AESU Dn 4	0	**AESU Done** Clears the AESU done status bit.	0 No action. 1 Clear status bit.
— 3–2	0	Reserved. Write to zero for future compatibility.	
DEU Err 1	0	**DEU Error Interrupt** Clears the DEU error status bit.	0 No action. 1 Clear status bit.
DEU Dn 0	0	**DEU Done** Clears the DEU done status bit.	0 No action. 1 Clear status bit.

27.7.5 Polychannel

The following sections describe the registers used by the Polychannel to manage the channel traffic.

27.7.5.1 Fetch FIFO Enqueue Counter (FFEC)

FFEC — Fetch FIFO Enqueue Counter — Offset 0xC1500

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	FETCH_FIFO_ENQ_CNT															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FETCH_FIFO_ENQ_CNT															
Type	R/W															
Reset	0x0000															

The Fetch FIFO Enqueue Counter indicates the total number of descriptor addresses that are enqueued to the channel fetch FIFOs. If this counter is all 1s (0xFFFFFFFF), the next count causes it to roll over to all 0s and set the CISR[FFE_CNT] bit if it is enabled (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186).

27.7.5.2 Descriptor Finished Counter (DFC)

DFC — Descriptor Finished Counter — Offset 0xC1508

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	DESCR_FINISHED_CNT															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DESCR_FINISHED_CNT															
Type	R/W															
Reset	0x0000															

The Descriptor Finished Counter indicates the total number of descriptors that successfully completed processing. It does not count descriptors that halt due to error. If this counter reaches all 1s (0xFFFFFFFF), the next count causes it to roll over to all 0s and set the CISR[DF_CNT] bit if it is enabled (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186).

27.7.5.3 Data Bytes In Counter (DBIC)

DBIC — Data Bytes In Counter — Offset 0xC1510

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	DATA_BYTES_IN_CNT															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	DATA_BYTES_IN_CNT															
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	DATA_BYTES_IN_CNT															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DATA_BYTES_IN_CNT															
Type	R/W															
Reset	0x0000															

The DBIC indicates the total number of bytes written into a primary EU input FIFO. If other parcels, such as context or ICV, are placed in the input FIFO, they are not counted. For a secondary EU, data going only to the secondary EU (such as a hash-only region or authentication data) is counted. However, data is never counted twice by the same counter. If this counter reaches all 1s (0xFFFFFFFFFFFFFFFF), the next count causes it to roll over to all 0s and set the CISR[DI_CNT] if it is enabled. (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186).

If this counter is read by software in 32-bit increments, then the least significant 32 bits must be read first, followed by the most significant 32 bits. If this counter is written by software in 32-bit increments, then the most significant 32 bits must be written first, followed by the least significant 32 bits.

Note: 32 bit reads and writes must not be interleaved, that is, a read low-write low-read high-write high sequence is not allowed. These restrictions are required to maintain counter coherency.

27.7.5.4 Data Bytes Out Counter (DBOC)

DBOC — Data Bytes Out Counter — Offset 0xC1518

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	DATA_BYTES_OUT_CNT															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	DATA_BYTES_OUT_CNT															
Reset	0x0000															
R/W	R/W															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	DATA_BYTES_OUT_CNT															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DATA_BYTES_OUT_CNT															
Type	R/W															
Reset	0x0000															

The DBOC indicates the total number of payload bytes read from a primary EU input FIFO. If other parcels, such as context or ICV, are read from the output FIFO, they are not counted. In no case is data ever counted twice by the same counter. If this counter reaches all 1s (0xFFFFFFFFFFFFFFFF), the next count causes it to roll over to all 0s and set the CISR[DO_CNT] if it is enabled. (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186).

If this counter is read by software in 32-bit increments, then the least significant 32 bits must be read first, followed by the most significant 32 bits. If this counter is written by software in 32-bit increments, then the most significant 32 bits must be written first, followed by the least significant 32 bits.

Note: 32 bit reads and writes must not be interleaved, that is, a read low-write low-read high-write high sequence is not allowed. These restrictions are required to maintain counter coherency.

27.7.6 Channel Registers and Structures

The following sections describe the registers and structures used by the four channels.

27.7.6.1 Channel Configuration Registers for Channels 1–4 (CCR[1–4])

CCR1 Channel Configuration Registers Offset 0xC1108 (0xC0108)
CCR2 Offset 0xC1208 (0xC0208)
CCR3 Offset 0xC1308 (0xC0308)
CCR4 Offset 0xC1408 (0xC0408)

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—													NPR	CON	R
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	FCC	WGN	PBS	—			BS	IWSE	—	EAE	CDWE	AWSE	NT	CDIE	—
Type	R/W															
Reset	0x0000															

Each CCR contains operational bits that permit configuration of the respective channel. The default address offset is used if the MCR[RCAn] bit is cleared (0) for the respective channel. The alternate address offset (shown in parentheses) is used if the MCR[RCAn] bit is set (1) for the respective channel. **Table 27-66** describes each field of the CCR.

Table 27-66. CCR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63–35	0	Reserved. Write to zero for future compatibility.	
NPR 34	0	**No Pop Reset** Used to perform a partial reset that does not clear the lower half of the CCR or the Fetch FIFO. After the reset sequence completes, this bit automatically clears (0) and the channel resumes operation by re-fetching the previous descriptor. This permits debug of a descriptor in-place whithout having to rewrite the descriptor pointer into the Fetch FIFO. The following conditions apply: • If the NPR bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request. • If the NPR bit is set after the channel has reserved one or more EUs, the channel requests a write from the controller to set the software reset bit of each reserved EU. The channel then releases the EU(s).	0 No special action. 1 Causes the same channel reset actions as CON, but the Fetch FIFO is left unchanged.

Table 27-66. CCR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
CON 33	0	**Continue** Used to perform a soft reset that does not clear the Fetch FIFO or the lower half of the CCR. After the reset sequence completes, this bit automatically clears (0) and the channel resumes normal operation servicing the next descriptor pointer in the Fetch FIFO, if any is present. The following conditions apply: • If the CON bit is set whil the channel is requesting an EU from the controller, the channel cancels its request. • If the CON bit is set after the channel is assigned one or more EUs, the channel requests a write from the controller to set the software reset bit of each reserved EU. The channel then releases the EU(s).	0 No special action. 1 Partial reset.
R 32	0	**Reset Channel** Used to perform a soft reset of the channel. The details of the software reset actions depend upon what the channel is doing when the bit is set: • If the R bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request. • If the R bit is set after the channel is assigned to an EU, the channel requests a write from the controller to set the software reset bit of each reserved EU. The channel then releases the EU(s). AFter the reset sequence is complete, the channel returns to the idle state and the R bit automatically returns to 0 and the channel resumes normat operation.	0 No special action. 1 Initiates a soft reset of the channel, clearing all of its internal state.
— 31–15	0	Reserved. Write to zero for future compatibility.	
FCC 14	0	**Fast Clock Counting** This bit controls the changes the watchdog timer count rate.	0 Watchdog timer counts normally. 1 Watchdog timer uses an accelerated count to assist in functional testing.
WGN 13	0	**Watchdog Timer Go Now** Enables/disables the watchdog timer.	0 Watchdog timer disabled. 1 Watchdog timer enabled..
PBS 12	0	**Permit Byte Summing** This bit controls whether writes to the EU input FIFO and reads from the EU output FIFO are counted by the Data Byte Counters.	0 Bytes counted. 1 Bytes not counted.
— 11–9	0	Reserved. Write to zero for future compatibility.	
BS 8	0	**Burst Size** This bit determines the burst size used by the SEC to access long text-data parcels in main memory.	0 Burst size is 64 bytes. 1 Burst size is 128 bytes.
IWSE 7	0	**ICV Writeback Status Enable** If this bit is set and the descriptor calls for ICV comparison, then at completion of descriptor processing, the channel writes back to the descriptor header all the required writeback information: DONE, ICCR0, and ICCR1 fields.	0 No special action. 1 Write back the required data after processing if ICV comparison is required.
— 6	0	Reserved. Write to zero for future compatibility.	
EAE 5	0	**Extend Address Enable** Selects whether to use 32-bit or 36-bit addressing.	0 Channel address bus is 32 bits. 1 Channel address bus is 36 bits.

Table 27-66. CCR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
CDWE 4	0	**Channel Done Writeback Enable** Enables/disables writeback of the DONE field after descriptor processing is completed if the NT or DN bit is set in the descriptor header. When enabled, the core processor can poll the memory location of the original descriptor to determine if the processing for that descriptor is completed.	0 Channel Done writeback disabled. 1 Channel Done writeback enabled.
AWSE 3	0	**Always Writeback Status Enable** When set, enables the channel after completing descriptor processing to writeback all the writeback information: DONE, ICCR0, and ICCR1 fields. When set, this field overrides IWSE, which then has no effect.	0 No special action. 1 Always writeback DONE, ICCR0, and ICCR1 fields after descriptor processing.
NT 2	0	**Notification Type** Selects when the channel generates notification. Channel notification can take the form of an interrupt, modified header writeback, or both, depending on the settings of CDIE and CDWE.	0 Global. The channel generates a done notification (if enabled) at the end of each descriptor. 1 Selected. The channel generates channel done notification (if enabled) at the end of a descriptor only when the DN bit in the descriptor header is set.
CDIE 1	0	**Channel Done Interrupt Enable** When set, enables the generation of a channel done interrupt. Depending on the setting of the NT and DN bits, the channel sends an interrupt to the core processor at the end of every descriptor processing or at the end of descriptors for which the DN bit in the header is set. See **Section 27.4.4**, *Channel Interrupts,* on page 27-20 for details.	0 No action. 1 Clear status bit.
— 0	0	Reserved. Write to zero for future compatibility.	

Note: The done interrupt, done writeback, and status writeback does not occur if an EU produces and error interrupt to the channel. In particular, if the ICV check error interrupt is enabled in the EU (see the ICE bit in the EU Interrupt Mask Register), and the ICV check finds a mismatch, then it generates an error interrupt, but no done interrupt and no writebacks.

Various fields in this table are used together with the Descriptor Header Control Word (the upper 32 bits of the Descriptor Header, see **Section 27.7.1.2**) to determine writeback options (see **Table 27-67**) and done interrupt options (see **Table 27-68**).

Table 27-67. Writeback Options

CCR Field				Descriptor Header	Writeback Action for a Descriptor Completing Without Error
AWSE	**CDWE**	**IWSE**	**NT**	**DN**	
1	x	x	x	x	write back Header fields DONE, ICCR0, ICCR1
0	1	x	1	0	no write back performed
0	1	x	1	1	write back Header field DONE
0	1	x	0	x	write back Header field DONE
0	x	1	x	x	if the descriptor header indicated ICV checking in AESU, CRCU, KEU, STEU, or MDEU, then write back Header fields DONE, ICCR0, and ICCR1.

Table 27-68. Done Interrupt Options

CCR		Descriptor Header	Done Interrupt action by Channel to Controller for a Descriptor completing without error
NT	CDIE	DN	
x	0	x	never assert done interrupt
0	1	x	assert done interrupt
1	1	0	never assert done interrupt
1	1	1	assert done interrupt

27.7.6.2 Channel Status Registers (CSR[1–4])

CSR1 Channel Status Registers Offset 0xC1110 (0xC0110)
CSR2 Offset 0xC1210 (0xC0210)
CSR3 Offset 0xC1310 (0xC0310)
CSR4 Offset 0xC1410 (0xC0410)

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—	GET_STATE							—	PUT_STATE						
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—							MAIN_STATE								
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—			FF_LEVEL					—				PRD	SRD	PD	SD
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	DOF	SOF	MDTE	—		IDH	—	EUE	WDT	SGLM	RSI	RSG	—			
Type	R/W															
Reset	0x0000															

This register contains status fields and counters that provide status information regarding the channel processing of the current descriptor. Bits 15–4 are error statuses. When a channel error interrupt is generated, bits in this range that are set indicate the source of the error. The default address offset is used if the MCR[RCAn] bit is cleared (0) for the respective channel. The alternate address offset (shown in parentheses) is used if the MCR[RCAn] bit is set (1) for the respective channel. **Table 27-69** describes the CSR fields.

Table 27-69. CSR Bit Field Descriptions

Bits	Reset	Description	Settings
— 63	0	Reserved. Write to zero for future compatibility.	
GET_ STATE 62–56	0	**Get State Machine State** Reflects the state of the Get State Machine when it last went to sleep. For debug purposes only.	
— 55	0	Reserved. Write to zero for future compatibility.	
PUT_STATE 54–48	0	**Put State Machine State** Reflects the state of the Put State Machine when it last went to sleep. For debug purposes only.	
— 47–41	0	Reserved. Write to zero for future compatibility.	
MAIN_STATE 40–32	0	**Main State Machine State** Reflects the state of the Main State Machine when it last went to sleep. For debug purposes only.	
— 31–29	0	Reserved. Write to zero for future compatibility.	
FF_LEVEL 28–24	0	**Fetch FIFO Level** This 5-bit counter indicates how many pointers are currently stored in the Fetch FIFO.	
— 23–20	0	Reserved. Write to zero for future compatibility.	
PRD 19	0	**Primary Reset Done** Reflects the state of the reset done signal from the assigned primary EU.	0 Assigned primary EU reset done signal inactive. 1 Assigned primary EU reset done signal active. Reset sequence complete and EU is ready to accept data.
SRD 18	0	**Secondary Reset Done** Reflects the state of the reset done signal from the assigned secondary EU.	0 Assigned secondary EU reset done signal inactive. 1 Assigned secondary EU reset done signal active. Reset sequence complete and EU is ready to accept data.
PD 17	0	**Primary EU Done** Reflects the state of the done interrupt from the assigned primary EU.	0 Assigned primary EU done interrupt inactive. 1 Assigned primary EU done interrupt active. EU processing complete and EU is ready to provide output data.
SD 16	0	**Secondary EU Done** Reflects the state of the done interrupt from the assigned secondary EU.	0 Assigned secondary EU done interrupt inactive. 1 Assigned secondary EU done interrupt active. EU processing complete and EU is ready to provide output data.

Table 27-69. CSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
DOF 15	0	**Double Fetch FIFO Write Overflow Error** Set when the channel fetch FIFO is full, SOF is set, and another write is made to the fetch FIFO. When this bit is set, the channel stops and activates an error interrupt. The channel does not start again until a continue or reset is generated via the CCR. You can clear this bit by writing a 1 to this bit.	0 No error detected. 1 Error detected.
SOF 14	0	**Single Fetch FIFO Write Overflow Error** Set when the channel fetch FIFO is full and another write is made to the fetch FIFO. The channel sets this bit and activates an error interrupt. The channel continues processing, but the descriptor pointer is lost. The core processor must clear this bit by writing a 1 to it.	0 No error detected. 1 Error detected.
MDTE 13	0	**Master Data Transfer Error** Set when the channel receives an error from the master bus interface. If the SEC is the bus master and detects the error, the controller passes the error to the channel in use. The channel halts and activates an interrupt. The channel can only be restarted by writing a 1 to the CON or R bit in the CCR, or resetting the whole SEC.	0 No error detected. 1 Error detected.
— 12–11	0	Reserved. Write to zero for future compatibility.	
IDH 10	0	**Illegal Descriptor Header** Possible causes of this error include: • Invalid primary EU indicated by the op_0 field. • Invalid secondary EU indicated by the op_1 field. • Descriptor type field in descriptor header indicates secondary EU transaction when not in snoop mode.	0 No error detected. 1 Error detected.
— 9	0	Reserved. Write to zero for future compatibility.	
EUE 8	0	**EU Error** Set when an EU assigned to this channel generates an error interrupt. The error can also be reflected in the CISR. **Note:** This bit can only be cleared by clearing the error interrupt source in the assigned EU that caused it to set.	0 No error detected. 1 Error detected.
WDT 7	0	**Watchdog Timeout** Set when the main state machine stays asleep too long. The timer runs only after EUs have been reserved, and does not run if the primary EU is the RNGU or PKEU. The timeout interval is controlled by the FCC field of the Channel Configuration Register. The channel stops when this bit is set and restarts when this bit is cleared.	0 No error detected. 1 Error detected.
SGLM 6	0	**Scatter/Gather Length Mismatch** Set when the total data size covered by a gather link table does not match the total data size from the main descriptor. The channel stops when this bit is set and restarts when this bit is cleared.	0 No error detected. 1 Error detected.
RSI 5	0	**Raid Size Incorrect** Set when the channel receives a descriptor of type RAID_XOR with data sizes that are not permitted.	0 No error detected. 1 Error detected.

Table 27-69. CSR Bit Field Descriptions (Continued)

Bits	Reset	Description	Settings
RSG 4	0	**Raid Scatter Gather Error** Set when the channel receives a descriptor of type RAID_XOR along with a set j bit. **Note:** Use of scatter/gather cannot be used with RAID_XOR type descriptors.	0 No error detected. 1 Error detected.
— 3–0	0	Reserved. Write to zero for future compatibility.	

Table 27-70. G_STATE and S_STATE Field Values

Value	Gather State Machine:
0x0	GS_IDLE
0x1	GS_LOAD_POINTER
0x2	GS_LOAD_POINTER_DONE
0x3	GS_LOAD_NEXT_POINTER
0x4	GS_PROCESS_POINTER
0x5	GS_TRANS_BLOCK
0x6	GS_TRANS_BLOCK_DONE
0x7	GS_TRANS_BYTES
0x8	GS_TRANS_BYTES_DONE
0x9	GS_INC_PAIR_PTR
0xA	GS_UPDATE
0xB	GS_DONE
0xC	GS_ERROR
0xD	GS_RELOAD
0xE	GS_TRANS_INBOUND
0xF	GS_TRANS_INBOUND_DONE

Table 27-71. CHN_STATE Field Values

Value	Channel State
0x00	IDLE
0x01	PROCESS_HEADER
0x02	FETCH_DESCRIPTOR
0x03	CHANNEL_DONE
0x04	CHANNEL_DONE_IRQ
0x05	CHANNEL_DONE_WRITEBACK
0x06	CHANNEL_DONE_NOTIFICATION
0x07	CHANNEL_ERROR
0x08	REQUEST_PRI_CHA
0x09	INC_DATA_PAIR_POINTER
0x0A	DELAY_DATA_PAIR_UPDATE
0x0B	EVALUATE_DATA_PAIRS
0x0C	WRITE_RESET_PRI
0x0D	RELEASE_PRI_CHA
0x0E	WRITE_RESET_SEC

Table 27-71. CHN_STATE Field Values (Continued)

Value	Channel State
0x0F	RELEASE_SEC_CHA
0x10	PROCESS_DATA_PAIRS
0x11	WRITE_MODE_PRI
0x12	WRITE_MODE_SEC
0x13	WRITE_DATASIZE_PRI
0x14	DELAY_RNGA_DONE
0x15	WRITE_DATASIZE_SEC_SNOOPIN
0x16	TRANS_REQUEST_WRITE_SNOOPIN
0x17	DELAY_PRI_SEC_DONE
0x18	TRANS_REQUEST_WRITE
0x19	WRITE_KEY_SIZE
0x1A	WRITE_CHA_GO
0x1B	DELAY_PRI_DONE
0x1E	WRITE_DATASIZE_SEC_SNOOPOUT
0x1F	TRANS_REQUEST_READ_SNOOPOUT
0x20	DELAY_SEC_DONE
0x21	TRANS_REQUEST_READ
0x22	EVALUATE_RESET
0x23	RESET_WRITE_RESET_PRI
0x24	RESET_RELEASE_PRI_CHA
0x25	RESET_WRITE_RESET_SEC
0x26	RESET_RELEASE_SEC_CHA
0x27	RESET_CHANNEL
0x28	WRITE_DATASIZE_PRI_POST
0x29	RESET_RELEASE_ALL
0x2A	RESET_RELEASE_ALL_DELAY
0x2B	REQUEST_SEC_CHA
0x2C	WRITE_DATASIZE_SEC
0x2D	WRITE_ICV_SIZE
0x2E	WRITE_SEC_CHA_GO_SNOOPOUT
0x2F	WRITE_PRI_CHA_GO_SNOOPIN
0x30	WRITE_SEC_CHA_GO_SNOOPIN
0x31	DELAY_1CYCLE
0x33	TRANS_EXTENT_READ
0x34	TRANS_EXTENT3
0x35	TRANS_EXTENT4
0x36	XOR_WRITE_READ_REG
0x37	DELAY_SEC_DONE_TLS
0x38	MAC_TO_CIPHER
0x39	MAC_TO_CIPHER_DONE
0x3A	READ_PRI_STATUS
0x3C	READ_SEC_STATUS
others	Reserved

27.7.6.3 Current Descriptor Pointer Register (CDPR)

CDPR1	Current Descriptor Pointer Registers	Offset 0xC1140 (0xC0140)
CDPR2		Offset 0xC1240 (0xC0240)
CDPR3		Offset 0xC1340 (0xC0340)
CDPR4		Offset 0xC1440 (0xC0440)

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	This register is not reset.															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—												EPTR			
Type	R															
Reset	This register is not reset.															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CUR_DES_PTR_ADRS															
Type	R															
Reset	This register is not reset.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CUR_DES_PTR_ADRS															
Type	R															
Reset	This register is not reset.															

The CDPR contains the address of the descriptor that the channel is currently processing. The default address offset is used if the MCR[RCAn] bit is cleared (0) for the respective channel. The alternate address offset (shown in parentheses) is used if the MCR[RCAn] bit is set (1) for the respective channel. The bits in the CDPR perform the functions described in **Table 27-72**.

Table 27-72. CDPR Bit Field Descriptions

Bits	Reset	Description
— 63–36	0	Reserved. Write to zero for future compatibility.
EPTR 35–32	0	**Extended Pointer** Concatenated as the upper 4 bits of the pointer address when extended mode is selected (EAE is high—see **Table 27-66** for details).
CUR_DES_PTR_ADRS 31–0	0	**Current Descriptor Pointer Address** Pointer to the system memory location of the current descriptor. This field reflects the starting location in system memory of the descriptor currently loaded into the DB. This value is updated whenever the channel requests a fetch of a descriptor from the controller. The value from the Fetch FIFO is transferred to the current descriptor pointer register immediately after the fetch is completed. This address is used as the destination for writeback of the modified header, if header writeback notification is enabled.

27.7.6.4 Channel Fetch FIFO (CFF)

CFF1	Channel Fetch FIFOs	Offset 0xC1148 (0xC0148)
CFF2		Offset 0xC1248 (0xC0248)
CFF3		Offset 0xC1348 (0xC0348)
CFF4		Offset 0xC1448 (0xC0448)

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—												EPTR			
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	FETCH_ADR															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FETCH_ADR															
Type	W															
Reset	0x0000															

Each channel contains a FIFO to store a queue of descriptor pointers starting with the address of the first byte of descriptors to process. Typically, the core processor creates a descriptor in memory containing all relevant mode and location information for the SEC and then launches the descriptor by writing its address to the SEC Fetch FIFO. The Fetch FIFO can hold up to 24 descriptor pointers at a time. When the end of the current descriptor is reached, the descriptor pointed to by the next location in the Fetch FIFO is read to launch the next descriptor. The Fetch Address is written into the FIFO only if the write includes the least significant byte. If extended addressing mode is selected (the EAE bit is high—see **Table 27-66** for details), then the Extended Fetch Address must be written before or concurrent with the Fetch Address. Specifying a FETCH_ADRS of 0 causes the channel to generate an error and stop. The default address offset is used if the MCR[RCAn] bit is cleared (0) for the respective channel. The alternate address offset (shown in parentheses) is used if the MCR[RCAn] bit is set (1) for the respective channel. The bits in the CFF perform the functions described in **Table 27-72**.

Table 27-73. CFF Bit Field Descriptions

Bits	Reset	Description
— 63–36	0	Reserved. Write to zero for future compatibility.
EPTR 35–32	0	**Extended Pointer** Concatenated as the upper 4 bits of the fetch address when extended mode is selected (EAE is high—see **Table 27-66** for details).
FETCH_ADR 31–0	0	**Fetch Address** Pointer to the memory location for the descriptor that the core processor wants the SEC to fetch.

27.7.6.5 Channel Descriptor Buffer (DB)

The descriptor buffer (DB) consists of eight 8-byte registers (DB[0–7]) and contains the current descriptor being processed by the channel. These registers are read-only because the descriptor is always fetched from system memory.

	0 15	16	17 23	24 27	28 31	32 63
DB0	Header					Reserved
DB1	Length0	J0	Extent0	—	Eptr0	Pointer0
DB2	Length1	J1	Extent1	—	Eptr1	Pointer1
DB3	Length2	J2	Extent2	—	Eptr2	Pointer2
DB4	Length3	J3	Extent3	—	Eptr3	Pointer3
DB5	Length4	J4	Extent4	—	Eptr4	Pointer4
DB6	Length5	J5	Extent5	—	Eptr5	Pointer5
DB7	Length6	J6	Extent6	—	Eptr6	Pointer6

Figure 27-57. Descriptor Format

For more information about the fields in a descriptor, see **Section 27.7.1.1**, *Descriptor Structure*, on page 27-99.

Note: The DBs are located at the following locations:

Channel 1: Offsets: 0xC1180–0xC11BF (0xC0180–0xC01BF)
Channel 2: Offsets 0xC1280–0xC12BF (0xC0280–0xC02BF)
Channel 3: Offsets 0xC1380–0xC13BF (0xC0380–0xC03BF)
Channel 4: Offsets 0xC1480–0xC14BF (0xC0480–0xC04BF)

The default address offset is used if the MCR[RCAn] bit is cleared (0) for the respective channel. The alternate address offset (shown in parentheses) is used if the MCR[RCAn] bit is set (1) for the respective channel.

27.7.7 PKEU Registers

27.7.7.1 PKEU Mode Register (PKEUMR)

PKEUMR — PKEU Mode Register — Offset 0xCC000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—								ROUTINE							
Type	R/W															
Reset	0x0000															

PKEUMR specifies the internal PKEU routine to execute. PKEUMR is cleared when the PKEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated. **Table 27-74** describes the PKEUMR fields.

Table 27-74. PKEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–8	0	Reserved. Write to zero for future compatibility.	
ROUTINE 7–0	0	**Routine** Specifies the PKEU routine to run.	See **Table 27-79** for values. See **Section 27.6.1.11**, *PKEU Routines,* on page 27-28 for details.

Table 27-75. ROUTINE Value Definitions

ROUTINE Value	Name	Description
0x00	Reserved	
0x01	CLEARMEMORY	Clear memory
0x02	MOD_EXP	FP: Exponential mod N and deconvert from Montgomery format
0x03	MOD_R2MODN	FP: Compute Montgomery converter (R^2 mod N)
0x04	MOD_RRMODP	FP: Compute Montgomery converter for Chinese Remainder Theorem (R_nR_p mod N)
0x05	EC_FP_AFF_PTMULT	FP EC: Multiply scalar times point in affine coordinates

Table 27-75. ROUTINE Value Definitions (Continued)

ROUTINE Value	Name	Description
0x06	EC_F2M_AFF_PTMULT	F2m EC: Multiply scalar times point in affine coordinates
0x07	EC_FP_PROJ_PTMULT	FP EC: Multiply scalar times point in projective coordinates
0x08	EC_F2M_PROJ_PTMULT	F2m EC: Multiply scalar times point in projective coordinates
0x09	EC_FP_ADD	FP EC: Add two points in projective coordinates
0x0A	EC_FP_DOUBLE	FP EC: Double a point in projective coordinates
0x0B	EC_F2M_ADD	F2m EC: Add two points in projective coordinates
0x0C	EC_F2M_DOUBLE	F2m EC: Double a point in projective coordinates
0x0D	F2M_R2	F2m: Compute Montgomery converter (R^2 mod N)
0x0E	F2M_INV	F2m: Invert mod N
0x0F	MOD_INV	FP: Invert mod N
0x10	MOD_ADD	FP: Add mod N
0x20	MOD_SUB	FP: Subtract mod N
0x30	MOD_MULT1_MONT	FP: Multiply mod N in Montgomery format
0x40	MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format
0x50	F2M_ADD	F2m: Add mod N
0x60	F2M_MULT1_MONT	F2m: Multiply mod N in Montgomery format
0x70	F2M_MULT2_DECONV	F2m: Multiply mod N and deconvert from Montgomery format
0x80	RSA_SSTEP	FP: Exponentiate mod N (combines MOD_R2MODN, POLY_F2M_MULT1_MONT, and MOD_EXP)
0x1D	MOD_EXP_TEQ	FP: Exponentiate mod N and deconvert from Montgomery format with timing equalization
0x1E	RSA_SSTEP_TEQ	FP: Exponentiate mod N with timing equalization (combines MOD_R2MODN, EC_F2M_MULT1_MONT, and MOD_EXP_TEQ)
0xFF	SPK_BUILD	Build PK data structure (data structure used by all elliptic curve routines)

27.7.7.2 PKEU Key Size Register (PKEUKSR)

PKEUKSR PKEU Key Size Register Offset 0xCC008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0000															

The Key Size Register reflects the number of significant bytes to use from PKEU Parameter Memory E in performing modular exponentiation or elliptic curve point multiplication. The range of values for this register, when performing either modular exponentiation or elliptic curve point multiplication, is from 1 to 256. Specifying a key size outside of this range causes a key size error (KSE) in the PKEU Interrupt Status Register.

Table 27-76. PKEUKSR Field Descriptions

Name	Reset	Description	Settings
— 63–12	0	Reserved. Write to zero for future compatibility.	
Key Size 11–0	0	**Key Size** Specifies the size in bytes of the PKEU parameter memory E to use for modulator exponentiation or elliptic curve point multiplication.	1–256 **Note:** Any other value causes a KSE in the PKEUISR.

27.7.7.3 PKEU AB Size Register (PKEUABSR)

PKEUABSR — PKEU AB Size Register — Offset 0xCC040

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				AB Size											
Type	R/W															
Reset	0x0000															

The AB Size register represents the operand size for the specific operands whenever it is required. The unit of the value written into the AB Size is in bits, even though internally the PKEU imposes a 32-bit alignment. Any data beyond the number of bits in the AB Size register, either in A and B-RAM (operands) is ignored. No error checking is performed whether the operand sizes are greater than the prime modulus or the field size and this can result is a wrong result. In other words, it is assumed that operands are modulo reduced before being written into the PKEU. Hence, the AB Size must be less than or equal to Data Size for a correct result. If the AB Size register is modified during processing, an error is generated.

Table 27-77. PKEUABSR Field Descriptions

Name	Reset	Description
— 63–12	0	Reserved. Write to zero for future compatibility.
AB Size 11–0	0	**AB Size** Specifies the maximum size in bits of data to use in the A- and B-RAM.

27.7.7.4 PKEU Data Size Register (PKEUDSR)

PKEUDSR PKEU Data Size Register Offset 0xCC010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Data Size											
Type	R/W															
Reset	0x0000															

The PKEU Data Size Register specifies the size of the significant portion of the modulus or irreducible polynomial in bits. Any value written to this register that is a multiple of 32 bits (for example, 128 bits, 160 bits, and so on) is represented internally as the same value (128 bits, 160 bits, respectively). Any value written that is not a multiple of 32 bits (for example, 132 bits, 161 bits, and so on), is represented internally as the next larger 32-bit multiple (160 bits, 196 bits, respectively). This internal rounding up to the next 32-bit multiple is described for information only. The minimum size valid for all routines to operate properly is 33 bits (internally 128 bits). The maximum size to operate properly is 2048 bits. A value in bits larger than 2048 result in a data size error.

An illegal data size error is generated as follows:

- All non-ECC routines with a data size > 512 generate an illegal data size error.
- All ECC routines with a data size > 128 generate an illegal data size error.

An AB Size = 0 (either intentionally written, or by ignoring and not writing at all) generates an illegal size error, except for routines that do not require an A or B operand such as the CLEAR_MEM routine.

Table 27-78. PKEUDSR Field Descriptions

Name	Reset	Description
— 63–12	0	Reserved. Write to zero for future compatibility.
Data Size 11–0	0	**Data Size** Specifies the maximum size in bits of data used internally. The value must range between 97–2048. Any other value generates a data size error.

27.7.7.5 PKEU Reset Control Register (PKEURCR)

PKEURCR — PKEU Reset Control Register — Offset 0xCC018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

This register contains three reset options specific to the PKEU.

Table 27-79. PKEURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupt** Setting this bit causes PKEU interrupts signalling done and error to reset. It further resets the state of the PKEU Interrupt Status Register.	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Setting this reinitializes the PKEU to accept another request without forcing the Interrupt Mask Register to change.This module initialization includes execution of an initialization routine, completion of which is indicated by the RD bit in the PKEUSR (see **Section 27.7.7.6**, *PKEU Status Register (PKEUSR),* on page 27-213).	0 No reset 1 Reset most of PKEU
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the PKEU. When SR is cleared, the PKEU enters a routine to perform proper initialization of the parameter memories. The PKEUSR[RD] indicates when the initialization routine is complete (see **Section 27.7.7.6**, *PKEU Status Register (PKEUSR),* on page 27-213).	0 No reset 1 Full PKEU reset.

27.7.7.6 PKEU Status Register (PKEUSR)

PKEUSR — PKEU Status Register — Offset 0xCC028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—								I	Z	HALT	—		EI	DI	RD
Type	R															
Reset	0x0000															

This Status Register contains fields that reflect the internal state of the PKEU. The PKEU Status Register is read-only. Writing to this location results in address error being reflected in the PKEU Interrupt Status Register (PKEUISR).

Table 27-80. PKEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–8	0	Reserved. Write to zero for future compatibility.	
I 7	0	**Infinity** This bit reflects the state of the PKEU infinity detect bit when last sampled. Only specific instructions within routines cause infinity to be modified, . Therefore, use the value of this bit carefully.	0 Infinity not detected. 1 Infinity detected.
Z 6	0	**Zero** This bit reflects the state of the PKEU zero detect bit when last sampled. Only specific instructions within routines cause the Z bit to be modified. Therefore, use the value of this bit carefully.	0 Zero not detected. 1 Zero detected.
HALT 5	0	**Halt** Indicates whether the PKEU is halted due to an error. **Note:** Because the error causing the PKEU to stop operating can be masked before reaching the Interrupt Status Register, the PKEU Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 PKEU not halted 1 PKEU halted
— 4–3	0	Reserved. Write to zero for future compatibility.	
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 PKEU is not signaling error 1 PKEU is signaling error

Table 27-80. PKEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal as sampled by the controller Interrupt Status Register (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186)	0 PKEU is not signalling done. 1 PKEU is signalling done.
RD 0	0	**Reset Done** This status bit, when high, indicates that the PKEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.7.7 PKEU Interrupt Status Register (PKEUISR)

PKEUISR — PKEU Interrupt Status Register — Offset 0xCC030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	EVM	INV	IE	BE	CE	KSE	DSE	ME	AE	—					
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the PKEU Interrupt Mask Register is zero (see **Section 27.6.1.8**, *PKEU Interrupt Mask Register*, on page 27-27).

If the PKEU Interrupt Status Register is non-zero, the PKEU halts and the PKEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186). In addition, if the PKEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the PKEU Reset Control Register. The definition of each bit in the PKEU Interrupt Status Register is shown in **Table 27-81**.

Table 27-81. PKEUISR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
EVM 14	0	**Even Modulus Error** When set, indicates that an even modulus was supplied for a PK operation that requires an odd modulus.	0 No even modulus error detected. 1 Even modulus error.
INV 13	0	**Inversion Error** When set, indicates that the inversion routine has a zero operand.	0 No inversion error detected. 1 Inversion error.
IE 12	0	**Internal Error** An internal processing error was detected while the PKEU was operating. **Note:** This bit is set any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Mask Register or by resetting the PKEU.	0 No internal error detected. 1 Internal error.
BE 11	0	**Boot Error** When set, indicates either a boot (reset) sequence or that RAM locations are not reset correctly.	0 No boot error detected. 1 Boot error.
CE 10	0	**Context Error** A PKEU key register, the Key Size Register, the Data Size Register, or the Mode Register was modified while the PKEU was operating.	0 No error detected. 1 Context error.
KSE 9	0	**Key Size Error** A value outside the bounds 1–256 bytes was written to the PKEU Key Size Register.	0 No error detected. 1 Key size error.
DSE 8	0	**Data Size Error** A value outside the range 97–2048 bits was written to the PKEU Data Size Register.	0 No error detected. 1 Data size error.
ME 7	0	**Mode Error** An illegal value was detected in the Mode Register. **Note:** Writing to reserved bits in the Mode Register is the most likely source of this error	0 No error detected. 1 Mode error.
AE 6	0	**Address Error** An illegal read or write address was detected within the PKEU address space.	0 No address error detected. 1 Address error detected.
— 5–0	0	Reserved. Write to zero for future compatibility.	

27.7.7.8 PKEU Interrupt Mask Register (PKEUIMR)

PKEUIMR — PKEU Interrupt Mask Register — Offset 0xCC038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	EVM	INV	IE	BE	CE	KSE	DSE	ME	AE	—					
Type	R															
Reset	0x1000															

The PKEU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.6.1.7**, *PKEU Interrupt Status Register*, on page 27-27), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs, and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then, upon detection of an error, the PKEU Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Table 27-82. PKEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
EVM 14	0	**Even Modulus Error** Enables/disables interrupt generation.	0 Even modulus error interrupt enabled. 1 Even modulus error interrupt disabled.
INV 13	0	**Inversion Error** Enables/disables interrupt generation.	0 Inversion error interrupt enabled. 1 Inversion error interrupt disabled.
IE 12	1	**Internal Error** Enables/disables interrupt generation.	0 Internal error interrupt enabled. 1 Internal error interrupt disabled.
BE 11	0	**Boot Error** Enables/disables interrupt generation.	0 Boot error interrupt enabled. 1 Bootl error interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Error interrupt enabled. 1 Context error interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Error interrupt enabled. 1 Key size error interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Error interrupt enabled. 1 Data size error interrupt disabled.
ME 7	0	**Mode Error** Enables/disables interrupt generation.	0 Error interrupt enabled. 1 Mode error interrupt disabled.

Table 27-82. PKEUIMR Field Descriptions (Continued)

Name	Reset	Description	Settings
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Address error interrupt enabled. 1 Address error interrupt disabled.
— 5–0	0	Reserved. Write to zero for future compatibility.	

27.7.7.9 PKEU End_of_Message Register (PKEUEOMR)

PKEUEOMR PKEU End_of_ Message Register Offset 0xCC050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The End_of_message Register in the PKEU is used to indicate the start of a new computation. Writing to this register causes the PKEU to execute the function requested by the ROUTINE field, per the contents of the parameter memories listed below. This register has no data size, and during the write operation, the core processor data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated.

27.7.7.10 PKEU Parameter Memories

The PKEU uses four 4096-bit memories to receive and store operands for the arithmetic operations the PKEU is asked to perform. In addition, results are stored in one particular parameter memory. Data addressing within these memories is big-endian, that is, the most significant byte is stored in the lowest address.

27.7.7.10.1 PKEU Parameter Memory A

This 4096 bit memory is used typically as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For

elliptic curve routines, this memory is segmented into four 1024 bit memories, and is used to specify particular curve parameters and input values. The PKEU Parameter Memory A region comprises offset 0xC200 to 0xC27F.

27.7.7.10.2 PKEU Parameter Memory B

This 4096 bit memory is used typically as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented in to four 1024 bit memories, and is used to specify particular curve parameters and input values, as well as to store result values. The PKEU Parameter Memory B region comprises offset 0xC400 to 0xC47F.

27.7.7.10.3 PKEU Parameter Memory E

This 4096 bit memory is non-segmentable, and specifies the exponent for modular exponentiation, or the multiplier k for elliptic curve point multiplication. This memory space is write only; a read of this memory space causes an address error to be reflected in the PKEU Interrupt Status Register. The PKEU Parameter Memory E region comprises offset 0xCA00 to 0xCBFF.

27.7.7.10.4 PKEU Parameter Memory N

This 4096 bit memory is non-segmentable, and stores the modulus for modular arithmetic and F_p elliptic curve routines. For F_2m elliptic curve routines, this memory stores the irreducible polynomial. The PKEU Parameter Memory N region comprises offset 0xC800 to 0xC9FF.

27.7.8 DEU Registers

27.7.8.1 DEU Mode Register (DEUMR)

DEUMR — DEU Mode Register — Offset 0xC2000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—												CM		TS	ED
Type	R/W															
Reset	0x0000															

The DEU Mode Register contains 3 bits that are used to program DEU operation. The Mode Register is cleared when the DEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated.

Note: The lowest 8 bits in this register are controlled by the MODE0 field in the descriptor header.

Table 27-83. DEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–4	0	Reserved. Write to zero for future compatibility.	
CM 3–2	0	**CBC/ECB** Specifies whether the DEU operates in cipher-block-chaining (CBC) or electronic code book (ECB) mode.	00 ECB mode. 01 CBC mode. 10 CFB – 1 mode. 11 OFB – 1 mode.
TS 1	0	**Triple/Single DES** Specifies whether to use the triple or single DES algorithm.	0 Single DES. 1 Triple DES.
ED 0	0	**Encryption/Decryption** Specifies whether to encrypt or decrypt the data.	0 Perform decryption. 1 Perform encryption.

27.7.8.2 DEU Key Size Register (DEUKSR)

DEUKSR — DEU Key Size Register — Offset 0xC2008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0000															

This value indicates the number of bytes of key memory to use in encrypting or decrypting. If the DEU Mode Register is set for single DES, any value other than 8 bytes automatically generates a key size error in the DEU Interrupt Status Register. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1=K3) or 24 bytes (168 bits for 3-key triple DES) generates an error. Triple DES always uses K1 to encrypt, K2 to decrypt, K3 to encrypt.

Table 27-84. PKEUKSR Field Descriptions

Name	Reset	Description	Settings
— 63–12	0	Reserved. Write to zero for future compatibility.	
Key Size 11–0	0	**Key Size** Specifies the size in bytes of the key memory to use for encrypitng or decrypting.	0x08 8 bytes (only legal value for single DES) 0x10 16 bytes (for 2-key 3DES, K1 = K3) 0x18 24 bytes (for 3-key 3DES) **Note:** Any other value causes a KSE in the DEUISR.

27.7.8.3 DEU Data Size Register (DEUDSR)

DEUDSR — DEU Data Size Register — Offset 0xC2010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Data Size											
Type	R/W															
Reset	0x0000															

This register stores the number of bits in the final message with an upper bound of 4096. Whatever number is written (and whatever truncated value is stored) must be a multiple of 64. All data to be processed by the DEU must be a multiple of the DES algorithm block size of 64 bits; the DEU does not automatically pad messages out to 64-bit blocks. If a data size that is not a multiple of 64 bits is written, a data size error is generated. Only the least significant 6 bits are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register. This register is cleared when the DEU is reset or reinitialized.

Table 27-85. DEUDSR Field Descriptions

Name	Reset	Description
— 63–12	0	Reserved. Write to zero for future compatibility.
Data Size 11–0	0	**Data Size** Specifies the maximum size in bits of data used internally. The value must range between 97–2048. Any other value generates a data size error.

27.7.8.4 DEU Reset Control Register (DEURCR)

DEURCR — DEU Reset Control Register — Offset 0xC2018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

This register allows three levels reset for the DEU, as defined by the three self-clearing bits described in **Table 27-86**.

Table 27-86. DEURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupt** Setting this bit causes DEU interrupts signalling done and error to reset. It further resets the state of the DEU Interrupt Status Register.	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Setting this bit perform like a software reset except that the Interrupt Mask Register remains unchanged. The module initialization includes execution of an initialization routine, completion of which is indicated by the RD bit in the DEUSR (see **Section 27.7.8.5**, *DEU Status Register (DEUSR),* on page 27-223).	0 No reset 1 Reset most of DEU.
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the DEU. All registers and internal states are returned to the defined reset state. When SR is cleared, the DEU enters a routine to perform proper initialization of the parameter memories. The DEUSR[RD] indicates when the initialization routine is complete (see **Section 27.7.8.5**, *DEU Status Register (DEUSR),* on page 27-223).	0 No reset 1 Full DEU reset.

27.7.8.5 DEU Status Register (DEUSR)

DEUSR — DEU Status Register — Offset 0xC2028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—								OFL							
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IFL								—		HALT	—		EI	DI	RD
Type	R															
Reset	0x0000															

This Status Register contains 6 fields that reflect the state of DEU internal signals. The DEU Status Register is read-only. Writing to this location results in an address error being reflected in the DEU Interrupt Status Register (DEUISR).

Table 27-87. DEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–24	0	Reserved. Write to zero for future compatibility.	
OFL 23–16	0	**Output FIFO Length** Indicates the number of 8-byte sets currently in the output FIFO.	
IFL 15–8	0	**Input FIFO Length** Indicates the number of 8-byte sets currently in the input FIFO.	
— 7–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the DEU is halted due to an error. **Note:** Because the error causing the DEU to stop operating can be masked before reaching the Interrupt Status Register, the DEU Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 DEU not halted 1 DEU halted
— 4–3	0	Reserved. Write to zero for future compatibility.	
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 DEU is not signaling error 1 DEU is signaling error

Table 27-87. DEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal as sampled by the controller Interrupt Status Register (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186)	0 DEU is not signalling done. 1 DEU is signalling done.
RD 0	0	**Reset Done** This status bit, when high, indicates that the DEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.8.6 DEU Interrupt Status Register (DEUISR)

DEUISR — DEU Interrupt Status Register — Offset 0xC2030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the DEU Interrupt Mask Register is zero (see **Section 27.6.2.7**, *DEU Interrupt Mask Register*, on page 27-43). If the DEU Interrupt Status Register is non-zero, the DEU halts and the DEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186). In addition, if the DEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the DEU Reset Control Register (see **Section 27.7.8.4**, *DEU Reset Control Register (DEURCR)*, on page 27-222). The definition of each bit in the DEU Interrupt Status Register is listed in **Table 27-88**.

Table 27-88. DEUISR Field Descriptions

Name	Reset	Description	Settings
— 63–14	0	Reserved. Write to zero for future compatibility.	
KPE 13	0	**Key Parity Error** If set, defined parity bits in the keys written to the key registers do not reflect odd parity correctly. **Note:** Key register 2 and key register 3 are checked for parity only if the appropriate DEU Mode Register bit indicates triple DES. Also, key register 3 is checked only if key size reg = 24. Key register 2 is checked only if key size reg = 16 or 24.	0 No key parity error detected. 1 Key parity error.
IE 12	0	**Internal Error** Indicates whether an internal processing error was detected while the DEU was processing.	0 No internal error detected. 1 Internal error.
ERE 11	0	**Early Read Error** Indicates whether the DEU IV register was read while the DEU was performing encryption.	0 No early read error detected. 1 Early read error.
CE 10	0	**Context Error** If set, indicates that DEU key register, the Key Size Register, Data Size Register, Mode Register, or IV register was modified while DEU was performing encryption.	0 No context error detected. 1 Context error.
KSE 9	0	**Key Size Error** IF set, indicates that an inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for triple DES) was written to the DEU Key Size Register.	0 No key size error detected. 1 Key size error.
DSE 8	0	**Data Size Error** If set, a value was written to the DEU Data Size Register that is not a multiple of 64 bits.	0 No data size error detected. 1 Data size error.
ME 7	0	**Mode Error** If set, an illegal value was detected in the Mode Register.	0 No mode error detected. 1 Mode error.
AE 6	0	**Address Error** If set, an illegal read or write address was detected within the DEU address space.	0 No address error detected. 1 Address error detected.
OFE 5	0	**Output FIFO Error** If set, the DEU output FIFO was detected non-empty upon write of DEU Data Size Register.	0 No output FIFO error detected. 1 Output FIFO error.
IFE 4	0	**Input FIFO Error** If set, the DEU input FIFO was detected non-empty upon generation of a done interrupt.	0 No input FIFO error detected. 1 Input FIFO error.

Table 27-88. DEUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
IFU 3	1	**Input FIFO Underflow** If set, the DEU input FIFO was read while empty.	0 No input FIFO underflow error detected. 1 Input FIFO underflow error.
IFO 2	0	**Input FIFO Overflow** If set, the DEU input FIFO was pushed while full. **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through core processor-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
OFU 1	0	**Output FIFO Underflow** If set, the DEU output FIFO was read while empty.	0 No output FIFO underflow error detected. 1 Output FIFO underflow error.
OFO 0	0	**Output FIFO Overflow** If set, the DEU output FIFO was pushed while full.	0 No output FIFO overflow error detected. 1 Output FIFO overflow error.

27.7.8.7 DEU Interrupt Mask Register (DEUIMR)

DEUIMR — DEU Interrupt Mask Register — Offset 0xC2038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
Type	R															
Reset	0x3000															

The Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.6.2.6**, *DEU Interrupt Status Register*, on page 27-43), if the corresponding bit in this register is set, then the error is ignored; no bit is set in the DEU Interrupt Status Register, and no error interrupt occurs. If the corresponding bit is not set, then upon detection of an error, the

Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Table 27-89. DEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–14	0	Reserved. Write to zero for future compatibility.	
KPE 13	1	**Key Parity Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IE 12	1	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 11	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ME 7	0	**Mode Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFE 5	0	**Output FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFE 4	0	**Input FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFU 3	0	**Input FIFO Underflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFU 1	0	**Output FIFO Underflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFO 0	0	**Output FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.

27.7.8.8 DEU End_of_Message Register (DEUEOMR)

DEUEOMR — DEU End_of_ Message Register — Offset 0xC2050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

Writing the end-of-message register is a handshake mechanism signalling that no more data is written to the input FIFO. DESA signals a done interrupt once the input FIFO is detected empty any time following the write to End-Of-Message. After the final message block is written to the input FIFO, the End_of_Message Register must be written. The value in the Data Size Register is used to determine how many bits of the final message block (always 64) is processed. Note that the end_of_ register has no data size, and during the write operation, the core processor data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned and no error is generated. Writing to this register is merely a trigger causing the DEU to process the final block of a message, allowing it to signal done interrupt.

27.7.8.9 DEU IV Register (DEUIVR)

For CBC mode, the initialization vector is written to and read from the DEU IV register. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading this memory location while the module is processing data generates an error interrupt.

Note: The DEUIVR is located at offset 0xC2100.

27.7.8.10 DEU Key Registers (DEUKR[1–3])

The DEU uses three write-only key registers, K1, K2, and K3, to perform encryption and decryption. In Single DES mode, only K1 can be written. The value written to K1 is simultaneously written to K3, auto-enabling the DEU for 112-bit Triple DES if the Key Size Register indicates 2 key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit Triple DES, K1 must be written first, followed by the write of K2, then K3.

Note: The DEU key registers are located at the following offsets:
DEUKR1 = Offset 0xC2400.
DEUKR2 = Offset 0xC2408
DEUKR3 = Offset 0xC2410
The Reading any of these memory locations generates an address error interrupt.

27.7.8.11 DEU FIFOs

DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the DEU FIFO address space enqueues data to the DEU input FIFO, and a read from anywhere in the DEU FIFO address space dequeues data from the DEU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, 4 bytes, or 8 bytes. When all 8 bytes of the staging register are written, the entire 8-byte set is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. Since the DEU data length should always be a multiple of 8 bytes, the last write should complete the 8-byte set. However, if there is any partial data set in the staging register when the DEU End_of_Message Register is written, the partial data is automatically padded with zeros to a full 8 bytes and enqueued to the input FIFO.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, that set is automatically dequeued from the FIFO so that the next 8 bytes (if any) become available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflow caused by reading or writing the DEU FIFOs are reflected in the DEU Interrupt Status Register.

Note: The DEU FIFOs occupy a memory space in the range defined by offsets 0xC2800–0xC2FFF

27.7.9 AESU Registers

27.7.9.1 AESU Mode Register (AESUMR)

AESUMR — AESU Mode Register — Offset 0xC4000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		SCM			—			ECM		AUX2	AUX1	AUX0	CM		ED
Type	R/W															
Reset	0x0000															

The AESU Mode Register contains 7 bit fields used to program the AESU. The Mode Register is cleared when the AESU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated. **Table 27-90** describes AESU Mode Register fields.

Note: The lowest 8 bits in this register are controlled by the MODE0 field in the descriptor header.

Table 27-90. AESUMR Field Descriptions

Name	Reset	Description	Settings
— 63–14	0	Reserved. Write to zero for future compatibility.	
SCM 13–11	0	**Sub-Cipher Mode** Specifies the number of sources to be XORd together for specific cipher modes.	*XOR Cipher Mode*: 1–6 are valid. All other values reserved. *Other cipher modes*: 0 is valid. All other values reserved.
— 10–8	0	Reserved. Write to zero for future compatibility.	
ECM 7–6	0	**Extend Cipher Mode** Used in combination with the CM field to define the AES operating mode.	See **Table 27-91** for details.

Table 27-90. AESUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
AUX2 5	0	**AUX2 Mode** The definition depends on the value of the four CM and ECM bits: • ECM = 1x and CM = 0x. Finalize MAC for CCM and GCM modes. • ECM = 10 or 01 and CM = 10. Use ICV bit value. • ECM = 00 and CM = 01. Enable RBP.	*If ECM = 1x and CM = 0x*: 0 Do not generate final MAC tag. 1 Generate final MAC tag after CCM/GCM processing is complete. For GCM, if message processing is split into multiple descriptions, AUX1 must equal 1 when AUX2 = 1. *If ECM = 10 or 01 and CM = 10*: 0 XCBC-MAC or CMAC cipher mode if ICV = 0. 1 XCBC-MAC with ICV or CMAC with ICV cipher mode if ICV = 1. *If ECM = 00 and CM = 01*: 0 CBC cipher mode. 1 CBC-RBP cipher mode.
AUX1 4	0	**AUX1 Mode** The definition depends on the value of the four CM and ECM bits: • ECM = 10 and CM = 00. Initialize CCM. • ECM = 10 and CM = 01. Generate final GHASH bit. Enables completion of GHASH computation by signaling that the last iteration of GHASH should be performed. This last iteration performs XOR of the current (intermediate) GHASH result with the concatenation of AAD and ciphertext bit lengths in case of GHASH(H, AAD, ciphertext) or with the concatenation of 064 and the bit length of IV in case of GHASH(H, {}, IV). As an exception, clear this bit if the whole message (IV+AAD+textdata) together with the generation of the final MAC is processed with one descriptor because in that case the generation of final GHASH is implied. Whenever AUX1 = 1 in GCM cipher mode, the total bit lengths of AAD, textdata or IV, must be provided in AESU Context Registers 9–10. • ECM = 10 and CM = 10. Use Context for XCBC-MAC derived keys. • ECM = 01 and CM = 10. Use Context for CMAC derived keys.	*If ECM = 10 and CM = 00*: 0 Do not initialize; context loaded by core processor. 1 Initialize new message with nonce/initialization vector. *If ECM = 10 and CM = 01*: 0 Do not perform the laster iteration in GHASH(H,AAD,ciphertext) or GHASH(H,{},IV) unless the message is processed and the final MAC computed in 1 descriptor. 1 Generate the final result of GHASH(H, AAD, ciphertext) or GHASH(H, {}, IV). Setting this bit implies that the message processing is split into multiple descriptors. *If ECM = 10 and CM = 10*: 0 Compute K1=E(K,16(01)), K2=E(K,16(02)), K3=E(K,(03)) and write K1 to Context Registers 3–4, K2 to Context Registers 5–6, and K3 to Context Registers 7–8. 1 Load keys: K1=[Key Data Registers 1–2], K2=[Key Data Registers 5–6], K3=[Key Data Registers 7–8]. *If ECM = 01 and CM = 10*: 0 Compute E(K,0^{128}) and write it to Context Registers 3–4. 1 Load E(K,0^{128}) and preserve it in Context Registers 3–4.

Table 27-90. AESUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
AUX0 3	0	**AUX0 Mode** The definition depends on the value of the four CM and ECM bits and the ED bit: • ED = 1 and ECM = 10 and CM = 01. Specifies GHASH mode. • ED = 1 and ECM = 10 or 01 and CM = 10. Specifies XCBC-MAC or CMAC Cipher mode.	*If ED = 1 and ECM = 10 and CM = 01*: 0 Perform GCM encryption. 1 Compute GHASH(H, AAD, ciphertext). *If ED = 1 and ECM = 10 or 01 and CM = 10*: 0 Generate final MAC tag; that is, XOR the final data block with K2/K3 (XCBC-MAC) or K1/K2 (CMAC) before encryption. 1 Do not generate final MAC tag before encryption to allow the message processing to be interrupted on the block boundary and later continue after a context switch.
CM 2–1	0	**Cipher Mode** Used in combination with the ECM field to define the AES operating mode.	See **Table 27-91** for details.
ED 0	0	**Encryption/Decryption** Specifies whether to encrypt or decrypt the data. **Note:** This bit is ignored in CTR, SRT, CMAC, and XCBC-MAC Cipher Modes.	0 Perform decryption. 1 Perform encryption.

Table 27-91. AES Cipher Modes

Mode	ECM	AUX2	CM
ECB	00	x	00
CBC	00	0	01
CBC-RBP	00	1	01
OFB	00	x	10
CTR	00	x	11
XTS	01	x	01
CMAC	01	0	10
CMAC with ICV	01	1	10
SRT	01	x	11
CCM	10	1	00
GCM	10	1	01
XCBC-MAC	10	0	10
XCBC-MAC with ICV	10	1	10
CFB128	10	x	11
CCM with ICV	11	1	00
GCM with ICV	11	1	01
XOR	11	x	11
Reserved	all others		

Notes: **1.** x indicates the value can be 1 or 0.
2. SRT is not a new AES mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0 srtp. See **Section 27.6.3.9.3**, *Context for SRT Cipher Mode,* on page 27-50 for more information on how SRT mode reduces context loading overhead.

27.7.9.2 AESU Key Size Register (AESUKSR)

AESUKSR — AESU Key Size Register — Offset 0xC4008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0000															

The AESU Key Size Register stores the number of bytes in the key (16, 24, 32). Any key data beyond the number of bytes in the Key Size Register is ignored. This register is cleared when the AESU is reset or reinitialized. If a key size other than 16, 24, or 32 bytes is specified, an illegal key size error is generated. If the Key Size Register is modified during processing, a context error is generated.

Table 27-92. AESUKSR Field Descriptions

Name	Reset	Description	Settings
— 63–12	0	Reserved. Write to zero for future compatibility.	
Key Size 11–0	0	**Key Size** Specifies the size in bytes of the key memory to use for encrypitng or decrypting.	0x10 16 bytes 0x18 24 bytes 0x20 32 bytes **Note:** Any other value causes a KSE in the AESUISR.

27.7.9.3 AESU Data Size Register (AESUDSR)

AESUDSR — AESU Data Size Register — Offset 0xC4010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—													Data Size		
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Data Size															
Type	R/W															
Reset	0x0000															

The AESU Data Size Register stores the number of bits of plaintext/ciphertext in the current descriptor. The number of Data Size Register bits used by the SEC, and the acceptable values for these bits, vary depending on the AES cipher mode selected. This register is cleared when the AESU is reset or reinitialized. Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

Table 27-93. AEUDSR Field Descriptions

Name	Reset	Description
— 63–19	0	Reserved. Write to zero for future compatibility.
Data Size 18–0	0	**Data Size** Stores the number of bits in the final message block. The required value is mode dependent. See **Table 27-94** for a list of bits used by selected cipher mode. **Note:** An improper data size sets AESUISR[DSE] to indicate a data size error. Modification of the data size during processing results in the setting of AESUISR[CE] bit to indicate a context error. AES-CCM mode does not support zero length AAD and zero length payload.

Table 27-94. Use of Data Size Register

AESU Cipher Mode	Register bits Used (others ignored)	Legal Values in Bits
ECB, CBC	lowest 7 bits	must be a multiple of 128
CCM		must be a multiple of 8
OFB, CMAC, CTR, SRT, CBC-RBP, XCBC-MAC, CFB128	all bits	
XTS		must be multiple of 8, minimum of 128
GCM		any value
XOR		must be a multiple of 256

27.7.9.4 AESU Reset Control Register (AESURCR)

AESURCR — AESU Reset Control Register — Offset 0xC4018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

This register allows three levels reset of just AESU, as defined by the three self-clearing bits:

Table 27-95. AESURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupt** Setting this bit causes AESU interrupts signaling done and error to reset. It further resets the state of the AESU Interrupt Status Register (AESUISR).	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Setting this bit perform like a software reset except that the Interrupt Mask Register remains unchanged. The module initialization includes execution of an initialization routine, completion of which is indicated by the RD bit in the AESUSR (see **Section 27.7.9.5**, *AESU Status Register (AESUSR),* on page 27-236).	0 No reset 1 Reset most of AESU
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the AESU. All registers and internal states are returned to the defined reset state. When SR is cleared, the AESU enters a routine to perform proper initialization of the parameter memories. The AESUSR[RD] indicates when the initialization routine is complete (see **Section 27.7.9.5**, *AESU Status Register (AESUSR),* on page 27-236).	0 No reset 1 Full AESU reset.

27.7.9.5 AESU Status Register (AESUSR)

AESUSR — AESU Status Register — Offset 0xC4028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—								OFL							
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IFL								—		HALT	ICCR		EI	DI	RD
Type	R															
Reset	0x0000															

The AESU Status Register is a read-only register that reflects the state of six status outputs. Writing to this location results in an address error being reflected in the AESU Interrupt Status Register.

Table 27-96. AESUSR Field Descriptions

Name	Reset	Description	Settings
— 63–24	0	Reserved. Write to zero for future compatibility.	
OFL 23–16	1	**Output FIFO Length** The number of 8-byte sets currently in the output FIFO.	
IFL 15–8	0	**Input FIFO Length** Indicates the number of 8-byte sets currently in the input FIFO.	
— 7–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the RNGU is halted due to an error. **Note:** Because the error causing the RNGU to stop operating can be masked before reaching the Interrupt Status Register, the RNGU Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 RNGU not halted 1 RNGU halted
ICCR 4–3	0	**Integrity Check Comparison Result** A passed or failed result is generated only if the selected cipher mode is CCM with ICV comparison.	00 No integrity check performed. 01 Integrity check comparison passed. 10 Integrity check comparison failed. 11 Reserved.

Table 27-96. AESUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 AESU is not signaling error 1 AESU is signaling error
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal as sampled by the controller Interrupt Status Register (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186)	0 AESU is not signalling done. 1 AESU is signalling done.
RD 0	0	**Reset Done** This status bit, when high, indicates that the AESU has completed its reset sequence. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.9.6 AESU Interrupt Status Register (AESUISR)

AESUISR — AESU Interrupt Status Register — Offset 0xC4030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the AESU Interrupt Mask Register is zero (see **Section 27.6.3.7**, *AESU Interrupt Mask Register*, on page 27-47). If the AESU Interrupt Status Register is non-zero, the AESU halts and the AESU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186). In addition, if the AESU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4]),* on page 27-199) and generates a channel error interrupt to the

controller. If the Interrupt Status Register is written from the core processor, 1s in the written value are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the AESU Reset Control Register (AESURCR). The definition of each bit in the AESU Interrupt Status Register is listed in **Table 27-97**.

Table 27-97. AESUISR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** If set, indicates that an ICV check was performed and that the supplied ICV did not match the value computed by the AESU.	0 No error detected. 1 Integrity check error.
— 13	0	Reserved. Write to zero for future compatibility.	
IE 12	0	**Internal Error** Indicates whether an internal processing error was detected while the AESU was processing. **Note:** This bit is set any time an enabled error condition occurs. It can only be cleared by setting the corresponding bit in the AESUIMR or by resetting the AESU.	0 No internal error detected. 1 Internal error.
ERE 11	0	**Early Read Error** Indicates whether the AESU IV register was read while the AESU was processing.	0 No early read error detected. 1 Early read error.
CE 10	0	**Context Error** If set, indicates that AESU key register, the Key Size Register, Data Size Register, Mode Register, or IV register was modified while the AESU was processing.	0 No context error detected. 1 Context error.
KSE 9	0	**Key Size Error** IF set, indicates that an inappropriate value (not 16, 24, or 32) was written to the AESU Key Size Register.	0 No key size error detected. 1 Key size error.
DSE 8	0	**Data Size Error** If set, an improper value was written to the AESU Data Size Register. See **Section 27.6.3.3**, *AESU Data Size Register,* on page 27-46 for details.	0 No data size error detected. 1 Data size error.
ME 7	0	**Mode Error** If set, indicates that invalid data was written to a register or that a reserved mode bit was set.	0 Valid data. 1 Reserved or invalid mode selected.
AE 6	0	**Address Error** If set, an illegal read or write address was detected within the AESU address space.	0 No address error detected. 1 Address error detected.
OFE 5	0	**Output FIFO Error** If set, the AESU output FIFO was detected non-empty upon write of AESU Data Size Register.	0 No output FIFO error detected. 1 Output FIFO non-empty error.
IFE 4	0	**Input FIFO Error** If set, the DEU input FIFO was detected non-empty upon generation of a done interrupt.	0 No input FIFO error detected. 1 Input FIFO non-empty error.

Table 27-97. AESUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
— 3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** If set, the AESU input FIFO was pushed while full. **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through core processor-controlled access, the AESU cannot accept FIFO inputs larger than 256 bytes without overflowing.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
OFU 1	0	**Output FIFO Underflow** If set, the AESU output FIFO was read while empty.	0 No output FIFO underflow error detected. 1 Output FIFO underflow error.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.9.7 AESU Interrupt Mask Register (AESUIMR)

AESUIMR — AESU Interrupt Mask Register — Offset 0xC4038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Type	R															
Reset	0x1000															

The AESU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.6.3.6**, *AESU Interrupt Status Register*, on page 27-46), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon

detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing

Table 27-98. AESUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 13	0	Reserved. Write to zero for future compatibility.	
IE 12	1	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 11	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ME 7	0	**Mode Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFE 5	0	**Output FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFE 4	0	**Input FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFU 1	0	**Output FIFO Underflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.9.8 AESU ICV Size Register (AESUICVSR)

AESUICVSR — AESU ICV Size Register — Offset 0xC4040

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—									ICV_Size						
Type	R/W															
Reset	0x0000															

The AESU ICV size register is used in AES hashing modes CMAC, GCM, and XCBC-MAC to specify the number of significant bytes in the received MAC tag supplied in Context Registers 3–4 (GCM w/ICV). AES truncates the computed MAC in Context Registers 1–2 to the same number of significant bytes and write zeros in the remaining positions. Note that the received MAC can be padded to the full 16 bytes with any value (that is. not necessarily zeros) when written into Context Registers 3-4. Acceptable values for ICV size are 8, 12, and 16 bytes. All other sizes are interpreted as 16. CCM with ICV does not use the ICV Size Register.

27.7.9.9 AESU End_of_Message Register (AESUEOMR)

AESUEOMR — AESU End_of_ Message Register — Offset 0xC4050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The AESU End_of_Message Register is used to indicate that an AES operation can be completed. After the final message block is written to the input FIFO, the End_of_Message Register must be written. The value in the Data Size Register is used to determine how many bits of the final message block (always 128) to process. Writing to this register causes the AESU to process the final block of a message, allowing it to signal a done interrupt. A read of this register always returns a zero value.

27.7.9.10 AESU Context Registers (AESUCR[1–12])

AESUCR[1–12] AESU Context Registers 1–12 Offset 0xC4100 + x*11

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	CONTEXT															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CONTEXT															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CONTEXT															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CONTEXT															
Type	R/W															
Reset	0x0000															

There are twelve 64-bit context data registers that allow the core processor to read/write the contents of the context used to process the message. The context must be written prior to the key data. If the Context Registers are written during message processing, a context error is generated. All Context Registers are cleared when a hard or soft reset or initialization is performed.

The Context Registers must be read when changing context and restored to their original values to resume processing an interrupted message. The actual number of context registers used in an operation is protocol specific and can range from 2 to 12, However, even though the protocol does not use lower numbered registers, all of the context registers up to the highest number used must be read to retrieve context and all s must be written back to restore context. For example, CTR mode uses only Context Register 5–7 to hold the Counter and Counter Modulus Exponent. So, effectively, the user must read the four empty place holder Context Registers 1–4 in addition to the three Context Registers that are used. The contents of the empty Context Registers need not be preserved, but when restoring the CTR mode context, the empty registers must be filled with 32 bytes of zeros before writing the saved Counter and Counter Modulus Exponent.

Context should be loaded starting with the lower bytes in the lowest 64-bit Context Register.

The context registers are summarized by modes in **Table 27-99**, **Table 27-100**, and **Table 27-100**.

Table 27-99. AESU Context Registers for Confidentiality Modes

Context Register (address offset)	Cipher Mode Providing Only Confidentiality			
	CBC/ CBC-RBP/ OFB/ CFB128 (Section 27.6.3.9.1)	CTR (Section 27.6.3.9.2)	SRT (Section 27.6.3.9.3	XTS (Section 27.6.3.9.4)
1 (0xC4100)	IV *		Counter *	I *
2 (0xC4108)				sector size *
3 (0xC4110)	—	—	Counter Modulus *	—
4 (0xC4118)			—	
5 (0xC4120)	—	Counter *	—	—
6 (0xC4128)				
7 (0xC4130)	—	Counter Modulus Exponent *	—	—

Notes: **1.** Context Registers 8 through 12 are unused for these modes
2. * = Must be written at start of new message, except if zero.
3. —- = ignored.

Table 27-100. AESU Context Registers for Integrity Modes

Context Register (address offset)	Cipher Mode Providing Only Data Integrity		
	XCBC-MAC (Section 27.6.3.9.5)	GCM-GHASH (Section 27.6.3.9.6)	CMAC (Section 27.6.3.9.7)
1 (0xC4100)	Computed MAC	Computed MAC	Computed MAC
2 (0xC4108)			
3 (0xC4110)	Received MAC*	—	Received MAC*
4 (0xC4118)			
5 (0xC4120)	$E(K, 0^{128})$	—	Key 3
6 (0xC4128)			
7 (0xC4130)	—	$len(AAD)^T$	Key 2
8 (0xC4138)		—	
9 (0xC4140)	—	H	Key 1**
10 (0xC4148)			
11 (0xC4150)	—	$len(AAD)^C$	—

Notes: **1.** Context register 12 is unused for these modes.
2. * Used only in ICV mode. Must be written at start of new message for ICV checking.
3. ^{T} = length of total data (in bits)
4. ** = Output only, not reloaded
5. ^{C} = length of data processed with current descriptor (in bits)
6. — = ignored

Table 27-101. AESU Context Registers for Modes Providing Confidentiality and Integrity

Context Register (address offset)	Cipher Mode Providing Confidentiality and Integrity	
	CCM (Section 27.6.3.9.8)	**GCM (Section 27.6.3.9.9)**
1 (0xC4100)	IV* / MAC	Computed MAC
2 (0xC4108)		
3 (0xC4110)	Encrypted MAC** / Decrypted MAC / Encrypted Counter	Received MAC
4 (0xC4118)		
5 (0xC4120)	Counter*	Counter
6 (0xC4128)		
7 (0xC4130)	Counter Modulus Exponent* (header size/ MAC size)***	$len(AAD)^T$
8 (0xC4138)	—	$len(IV)^T$
9 (0xC4140)	—	Y_0
10 (0xC4148)		
11 (0xC4150)	—	$len(AAD)^C$
12 (0xC4158)		$len(IV)^C$

Notes:
1. * = Must be written at start of new message, except if zero
2. ** = Must be written at start of new CCM decryption
3. *** = The Header and MAC Sizes are internally constructed by the AES engine; then, that information is included inside Context Register 7 for context switching purposes
4. ^{C} = length of data processed with current descriptor (in bits)
5. ^{T} = length of total data (in bits)
6. ^{ICV} = Needed only in ICV mode
7. — = Ignored

See **Section 27.6.3.9**, *AESU Context Registers,* on page 27-47 for details on how to use and program the Context Registers.

27.7.9.11 AESU Key Registers (AESUK[U/L]R[1–3])

AESUKUR1	AESU Key Registers	Offset 0xC4400
AESUKLR1		Offset 0xC4408
AESUKUR2		Offset 0xC4410
AESUKLR2		Offset 0xC4418

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	KEY															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	KEY															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KEY															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	KEY															
Type	R/W															
Reset	0x0000															

The AESU key registers are organized as an 8-byte upper half following by an 8-byte lower half. They hold 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to KEY 1U. Any key data written to bytes beyond the value written to the Key Size Register are ignored. The Key Data Registers are cleared when the AESU is reset or reinitialized. If these registers are modified during message processing, a context error is generated.

27.7.9.12 AESU FIFOs

AESU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the AESU FIFO address space enqueues data to the AESU input FIFO, and a read from anywhere in the AESU FIFO address space dequeues data from the AESU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, 4 bytes, or 8 bytes. When all 8 bytes of the staging register are written, the entire set is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the AESU End_of_Message Register is written.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, that 8-bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflows caused by reading or writing the AESU FIFOs are reflected in the AESU Interrupt Status Register.

The AESU fetches data 128 bits at a time from the input FIFO. During processing, the input data is encrypted or decrypted with the key and initialization vector (CBC mode only) and the results are placed in the output FIFO. The output size is the same as the input size.

The input FIFO can be written any time the number of 8-byte sets currently in the input FIFO (as indicated by the IFL field of the AESU Status Register) is less than 32. There is no limit on the total number of bytes in a message. The number of bits in the final message block must be set in the Data Size Register.

The output FIFO can be read any time the OFR signal is asserted (as indicated in the AESU Status Register). The result indicates that the number of bytes in the output FIFO is at or above the threshold specified in the Mode Register.

For AES-CCM mode, the input data stream to the input FIFO consists of the CCM Header, followed by the Additional Authenticated Data (AAD), followed by Plaintext if encrypting, or Ciphertext if decrypting. This mode does not support zero length AAD and zero length payload. Note that AESU only supports 2 byte CCM headers and does not support extended CCM headers.

Note: The AESU FIFOs occupy a memory space in the range defined by offsets 0xC4800–0xC4FFF

27.7.10 MDEU Registers

27.7.10.1 MDEU Mode Register (MDEUMR)

MDEUMR — MDEU Mode Register (Old, NEW = 0) — Offset 0xC6000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—			MDEU_B	—		NEW=0	—	CONT	CICV	SMAC	INIT	HMAC	PD	ALG	
Type	R/W															
Reset	0x0000															

MDEUMR — MDEU Mode Register (New, NEW = 1)) — Offset 0xC6000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—			MDEU_B	—	STIB	NEW=1	—	CONT	CICV	SMAC	INIT	HMAC	EALG	ALG	
Type	R/W															
Reset	0x0000															

The MDEU Mode Register is used to program the function of the MDEU. The least significant 8 bits of this register are specified by the user through the MODE0 or MODE1 field of the descriptor header. The remaining bits are supplied by the channel and thus are not under direct user control. The MDEU Mode Register has two configurations, determined by the value of the

NEW bit. The new configuration (NEW = 1) is used only by TLS/SSL descriptor types (1000_1, 1001_1). The old configuration (NEW = 0) is used by all other descriptor types. The old configuration is the same as the one used in SEC 2.0, except for the CICV and SMAC bits. The Mode Register is cleared when the MDEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated. **Table 27-102** describes MDEU Mode Register fields.

Table 27-102. MDEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–13	0	Reserved. Write to zero for future compatibility.	
MDEU_B 12	0	**MDEU_B Alternate Algorithms** Selects which algorithms are enabled by the ALG bits.	0 MDEU-A group (SHA-1, SHA-256, MD5, and SHA-224) 1 MDEU-B group (SHA-384, SHA-256, SHA-512, and SHA-224)
— 11	0	Reserved. Write to zero for future compatibility.	
STIB 10	0	**SSL/TLS Inbound Block Cipher (New mode only)** When this bit is set, upon receiving End_of_message, the MDEU performs a calculation involving the last valid byte of data written into its input FIFO (which is Pad Length) to compute a final data size. The MDEU then processes the amount of data specified by this data size, and completes the message digest.	0 Normal operation. 1 Special operation for SSL/TLS inbound, block cipher only.
NEW 9	0	**New Mode** Determines which format the MDEU Mode Register uses.	0 Old configuration. 1 New configuration.
— 8	0	Reserved. Write to zero for future compatibility.	
CONT 7	0	**Continue** Most operations require this bit to be cleared. It is set only when the data to be hashed is spread across multiple descriptors.	0 Do auto padding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash is continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
CICV 6	0	**Compare Integrity Check Values** Indicates whether to do an ICV check. The number of bytes to be compared is given by the ICV Size Register. This is only applicable to descriptor types that provide for reading in an ICV value.	0 Normal operation; no ICV comparison. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU input FIFO. If the ICVs do not match, send an error interrupt to the channel.
SMAC 5	0	**SSL MAC Operation** Specifies whether to perform an SSL3.0 MAC operation. This requires a key and key length. **Note:** If this bit is set, HMAC should be 0.	0 Normal operation. 1 Perform.an SSL3.0 MAC operation
INIT 4	0	**Initialization** Indicates whether to initialize the MDEU. If initialization is not done, the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit must be 0 on all but the first descriptor.	0 Do not initialize digest registers. 1 Do an algorithm-specific initialization of the digest registers.

Table 27-102. MDEUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
HMAC 3	0	**HMAC Operation** Selects whether to perform an HMAC operation. The HMAC operation requires a key and a key length. If this bit is set, then SMAC should be cleared (0).	0 Normal operation. 1 Perform.an HMAC operation.
PD/EALG 2	0	**Padding** For Old configurations (NEW = 0), the value of this bit must be programmed to be the inverse of the CONT bit. **EALG** For New configurations (NEW = 1), this bit is an additional extension bit for the ALG field.	*For PD, New = 0*: 0 This hash is continued in a subsequent descriptor. Do not autopad and do not complete the message digest. 1 Do auto padding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. *For EALG, New = 1:* 0 Only valid value. 1 Reserved.
ALG 1–0	0	**Message Algorithm** Selects the message algorithm to use.	00 SHA-160 algorithm (full name for SHA-1). 01 SHA-256 algorithm. 10 MD5 algorithm. 11 SHA-224 algorithm.

27.7.10.2 MDEU Key Size Register (MDEUKSR)

MDEUKSR — MDEU Key Size Register — Offset 0xC6008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—								Key Size							
Type	R/W															
Reset	0x0000															

The MDEU Key Size Register value indicates the number of bytes of key memory that should be used in HMAC generation. MDEU supports at most one block of key. MDEU generates a key size error if the value written to this register exceeds 64 bytes for MD5, SHA-1, SHA-224, or SHA-256, or if the value exceeds 128 bytes for SHA-384 or SHA-512.

27.7.10.3 MDEU Data Size Register (MDEUDSR)

MDEUDSR — MDEU Data Size Register — Offset 0xC6010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—											Data Size				
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Data Size															
Type	R/W															
Reset	0x0000															

The MDEU Data Size Register indicates the number of bits of data to be processed. The Data Size field is a 21-bit signed number. Values written to this register are added to the current register value. Multiple writes are allowed. The MDEU processes data when there is a positive value in this register and there is data available in the MDEU input FIFO. (Negative values can arise in inbound processing, when it is necessary to hold back data from the MDEU until the pad length is decrypted.). Because the MDEU does not support bit offsets, the least significant 3 bits must be written as 0 and are always read as zero. Furthermore, when the CONT bit of the MDEU Mode Register is high, the data size must be a multiple of the block size (that is, 512 bits for MD5, SHA-1, SHA-224, and SHA-256; 1024 bits for SHA-384 and SHA-512). Violating either of these conditions causes a data size error (the MDEUIDR[DSE] bit is set).

This register is cleared when the MDEU is reset or reinitialized. At the end of processing, its contents are decremented down to zero (unless there is an error interrupt).

Note: Writing to the Data Size Register allows the MDEU to enter auto-start mode. Therefore, always write the required Context Registers prior to writing the data size.

27.7.10.4 MDEU Reset Control Register (MDEURCR)

MDEURCR — MDEU Reset Control Register — Offset 0xC6018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

The MDEU Reset Control Register allows three levels reset for the MDEU only, as described in **Table 27-103**.

Table 27-103. MDEURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupt** Setting this bit causes MDEU interrupts signalling done and error to reset. It further resets the state of the MDEU Interrupt Status Register.	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Module initialization is almost the same as a software reset except that the Interrupt Mask Register remains unchanged.	0 No reset 1 Reset most of MDEU
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the MDEU. All registers and internal states are returned to the defined reset state.	0 No reset 1 Full MDEU reset.

27.7.10.5 MDEU Status Register (MDEUSR)

MDEUSR — MDEU Status Register — Offset 0xC6028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—										HALT	ICCR		EI	DI	RD
Type	R															
Reset	0x0000															

The MDEUSR reflects the state of the MDEU internal signals. The majority of these internal signals reflect the state of low-level MDEU functions, such as data padding, key padding, and so forth., and are not important to the user, however the user should be aware that reads of this register especially during processing are likely to return non-zero values for many of the most significant 58 bits. The MDEUSR is read-only. Writing to this location results in an address error being reflected in the MDEU Interrupt Status Register (MDEUISR).

Table 27-104. MDEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the MDEU is halted due to an error. **Note:** Because the error causing the MDEU to stop operating can be masked before reaching the Interrupt Status Register, the MDEUISR is used to provide a second source of information regarding errors that prevent normal operation.	0 MDEU not halted 1 MDEU halted
ICCR 4–3	0	**Integrity Check Comparison Result** A passed or failed result is generated only if ICV comparison is enabled.	00 No integrity check performed. 01 Integrity check comparison passed. 10 Integrity check comparison failed. 11 Reserved.
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186).	0 MDEU is not signaling error 1 MDEU is signaling error

Table 27-104. MDEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal as sampled by the controller Interrupt Status Register (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186)	0 MDEU is not signalling done. 1 MDEU is signalling done.
RD 0	0	**Reset Done** This status bit, when high, indicates that the MDEU has completed its reset sequence. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.10.6 MDEU Interrupt Status Register (MDEUISR)

MDEUISR — MDEU Interrupt Status Register — Offset 0xC6030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—		IFO	—	
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the MDEU Interrupt Mask Register is zero (see **Section 27.7.11.7**, *AFEU Interrupt Mask Register (AFEUIMR),* on page 27-270).

If the MDEU Interrupt Status Register is non-zero, the MDEU halts and the MDEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186). In addition, if the MDEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4]),* on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s that are written in the value are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the MDEU Reset Control Register (MDEURCR[RI]). The definition of each bit in the MDEUISR is listed in **Table 27-105**.

Table 27-105. MDEUISR Field Descriptions

Name	Reset	Description	Settings
— 63–14	0	Reserved. Write to zero for future compatibility.	
ICE 13	0	**Integrity Check Error** If set, indicates that an ICV check was performed and that the supplied ICV did not match the value computed by the MDEU.	0 No error detected. 1 Integrity check error.
— 12	0	Reserved. Write to zero for future compatibility.	
IE 11	0	**Internal Error** Indicates whether the MDEU is locked and requires a reset before use. **Note:** This bit is set any time an enabled error condition occurs. It can only be cleared by resetting the MDEU.	0 No internal error detected. 1 Internal error.
ERE 10	0	**Early Read Error** Indicates whether the MDEU context was read before the MDEU completed the hashing operation.	0 No early read error detected. 1 Early read error.
CE 9	0	**Context Error** If set, indicates that MDEU key register, Key Size Register, or Data Size Register was modified while the MDEU was performing its hashing operation.	0 No context error detected. 1 Context error.
KSE 8	0	**Key Size Error** If set, indicates an error due to one of the following two causes: • A value greater than 64 bytes was written to the MDEU Key Size Register. • In either an HMAC or SMAC operation, the key size was not written prior to writing the data size or receiving a CHA_GO command.	0 No key size error detected. 1 Key size error.
DSE 7	0	**Data Size Error** If set, indicates that data with a size not a multiple of the block size (512 or 1024 depending on protocol) was found when the MDEUMR[CONT} bit was cleared (0).	0 No data size error detected. 1 Data size error.
ME 6	0	**Mode Error** If set, indicates one of the following conditions: • A reserved bit in the MDEUMR is set. • The ALG field of the MDEUMR contains an illegal value.	0 No error detected. 1 Mode error.
AE 5	0	**Address Error** If set, an illegal read or write address was detected within the MDEU address space.	0 No address error detected. 1 Address error detected.
— 4–3	0	Reserved. Write to zero for future compatibility.	

Table 27-105. MDEUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
IFO 2	0	**Input FIFO Overflow** If set, the MDEU input FIFO was pushed while full. **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through core processor-controlled access, the MDEU cannot accept FIFO inputs larger than 256 bytes without overflowing.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
— 1–0	0	Reserved. Write to zero for future compatibility.	

27.7.10.7 MDEU Interrupt Mask Register (MDEUIMR)

MDEUIMR — MDEU Interrupt Mask Register — Offset 0xC6038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—		IFO	—	
Type	R															
Reset	0x1000															

The MDEU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.6.4.7**, *MDEU Interrupt Status Register*, on page 27-70), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then

upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Table 27-106. MDEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–14	0	Reserved. Write to zero for future compatibility.	
ICE 13	0	**Integrity Check Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 12	0	Reserved. Write to zero for future compatibility.	
IE 11	0	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 10	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 9	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 8	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 7	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ME 6	0	**Mode Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
AE 5	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 4–3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
— 1–0	0	Reserved. Write to zero for future compatibility.	

27.7.10.8 MDEU ICV Size Register (MDEUICVSR)

MDEUICVSR — MDEU ICV Size Register — Offset 0xC6040

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—									ICV_Size						
Type	R/W															
Reset	0x0000															

The MDEU ICV size register stores the number of bytes of the ICV result to be compared if the MDEU performs ICV comparison (see **Section 27.6.4.2**, *MDEU Mode Register*, on page 27-68). This register is cleared when the MDEU is reset or reinitialized.

27.7.10.9 MDEU End_of_Message Register (MDEUEOMR)

MDEUEOMR — MDEU End_of_ Message Register — Offset 0xC6050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The End_of_message Register in the MDEU is used to indicate an authentication operation can be completed. After the final message block is written to the input FIFO, the End_of_Message Register must be written. The value in the Data Size Register is used to determine how many bits of the final message block (always 512) are processed. Note that this register has no data size, and during the write operation, the core processor data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal done interrupt.

27.7.10.10 MDEU Context Registers (MDEUCR)

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provides the resulting message digest or HMAC along with an aggregate bit count.

Note: All SHA algorithms are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU Mode Register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location while the module is not done generates an error interrupt.

After a power-on reset, all the MDEU Context Register values are cleared (0). The MDEU Context Registers are initialized if the INIT bit is set in the MDEU Mode Register. However, all registers are initialized, regardless of mode selected, but only the appropriate Context Register values are used in hash generation per the mode selected. Even though the user typically does not need to know about the MDEU Context Register initialization values; they are documented for completeness in the event that the user reads these registers using core processor-controlled access. MDEU resets via the MDEU Reset Control Register (**Section 27.6.4.5**, *MDEU Reset Control Register*, on page 27-70) or SEC global software reset (**Section 27.7.4.1**, *Master Control Register (MCR)*, on page 27-177) do not clear these registers. **Figure 27-58** shows the memory map and register contents for the various hash modes. If SHA-384 or SHA-512 is selected, then each of the registers A, B, C, D, E, F, G, and H is 64-bits, instead of the 32 bit-width used with the other hash algorithms. As a result, the base address for each context register is shifted to adjust for the difference in register size.

	63 — 32	31 — 0	Offset
MD-5	A = 0x01234567	B = 0x89ABCDEF	0xC_6100
SHA-1	A = 0x67452301	B = 0xEFCDAB89	
SHA-224	A = 0xC1059ED8	B = 0x367CD507	
SHA-256	A = 0x6A09E667	B = 0xBB67AE85	
SHA-384	A = 0xCBBB9D5DC1059ED8		
SHA-512	A = 0x6A09E667F3BCC908		
MD-5	C = 0xFEDCBA98	D = 0x76543210	0xC_6108
SHA-1	C = 0x98BADCFE	D = 0x10325476	
SHA-224	C = 0x3070DD17	D = 0xF70E5939	
SHA-256	C = 0x3C6EF372	D = 0xA54FF53A	
SHA-384	B = 0x629A292A367CD507		
SHA-512	B = 0xBB67AE8584CAA73B		
MD-5	E = 0xF0E1D2C3	F = 0x8C68059B	0xC_6110
SHA-1	E = 0xC3D2E1F0	F = 0x9B05688C	
SHA-224	E = 0xFFC00B31	F = 0x68581511	
SHA-256	E = 0x510E527F	F = 0x9B05688C	
SHA-384	C = 0x9159015A3070DD17		
SHA-512	C = 0x3C6EF372FE94F82B		
MD-5	G = 0xABD9831F	H = 0x19CDE05B	0xC_6118
SHA-1	G = 0x1F83D9AB	H = 0x5BE0CD19	
SHA-224	G = 0x64F98FA7	H = 0xBEFA4FA4	
SHA-256	G = 0x1F83D9AB	H = 0x5BE0CD19	
SHA-384	D = 0xh152fecd8hf70e5939		
SHA-512	D = 0xha54ff53ah5f1d36f1		
MD5, SHA1, SHA-224, SHA-256	Message Length Count = 0		0xC_6120
SHA-384	E = 0x67332667FFC00B31		
SHA-512	E = 0x510E527FADE682D1		
MD5, SHA1, SHA-224, SHA-256	reserved		0xC_6128
SHA-384	F = 0x8EB44A8768581511		
SHA-512	F = 0x9B05688C2B3E6C1F		
MD5, SHA1, SHA-224, SHA-256	reserved		0xC_6130
SHA-384	G = 0xDB0C2E0D64F98FA7		
SHA-512	G = 0x1F83D9ABFB41BD6B		
MD5, SHA1, SHA-224, SHA-256	reserved		0xC_6138
SHA-384	H = 0x47B5481DBEFA4FA4		
SHA-512	H = 0x5BE0CD19137E2179		
MD5, SHA1, SHA-224, SHA-256	reserved		0xC_6140
SHA-384, SHA-512	Message Length Count = 0		

Figure 27-58. MDEU Context Register Map

27.7.10.11 MDEU Key Registers (MDEUKR[1–8])

The MDEU maintains sixteen 64-bit registers for writing an HMAC key. Only the first eight registers are used for MD-5, SHA-1, SHA-224, or SHA-256. The IPAD and OPAD operations are performed automatically on the key data when required.

Note: All SHA algorithms are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU Mode Register indicates MD5 is the hash of choice.

Note: The MDEU key registers are located in a range defined by offsets 0xC6400–0xC647F.

27.7.10.12 MDEU Input FIFO

MDEU uses an input FIFO to hold data to be hashed (followed in some case by an ICV value for ICV checking). Normally, the channels control all access to this FIFO. For core processor-controlled operation, a write to anywhere in the MDEU FIFO address space enqueues data to the MDEU input FIFO, and a read from anywhere in this address space returns all zeros.

When the core processor writes to the MDEU FIFO (using core processor-controlled access), it can write to any FIFO address using byte, 4-byte, or 8-byte accesses. The MDEU assembles these bytes from left to right, that is, the first bytes written are placed in the most significant bit-positions. Whenever the MDEU accumulates 8 bytes, these 8 bytes are automatically enqueued into the FIFO, and any remaining bytes are left-justified in preparation for assembling the next 8 bytes. It is not necessary to fill all bytes of the final 8-byte set. Any remaining bytes in the staging register are automatically padded with zeros and forced into the input FIFO when the MDEU End_of_Message Register is written.

Overflows caused by writing the MDEU FIFO are reflected in the MDEU Interrupt Status Register.

Note: The MDEU input FIFO is located in a range defined by offsets 0xC6800–0xC6FFF.

27.7.11 AFEU Registers

27.7.11.1 AFEU Mode Register (AFEUMR)

AFEUMR — AFEU Mode Register — Offset 0xC8000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													CS	DC	PP
Type	R/W															
Reset	0x0000															

The AFEU Mode Register (AFEUMR) contains three bits used to program the AFEU. The Mode Register is cleared when the AFEU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the Mode Register is modified during processing, a context error is generated. **Table 27-107** describes AFEU Mode Register fields.

Table 27-107. AFEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
CS 2	0	**Context Source** If set, causes the context to be moved from the input FIFO into the S-box prior to starting encryption/decryption. Otherwise, context should be directly written to the Context Registers or context should be generated automatically via key permutation. The CS value is only checked if the PP bit is set.	0 Context not from FIFO. 1 Context from input FIFO.
DC 1	0	**Dump Context** If set, causes the context to be moved from the S-box to the output FIFO following assertion of the AFEU done interrupt.	0 Do not dump context. 1 After cipher, dump context.
PP 0	0	**Prevent Permute** Normally, AFEU receives a key and uses that information to randomize the S-box. If reusing a context from a previous descriptor, set this bit to prevent the AFEU from reperforming this permutation step.	0 Perform S-Box permutation. 1 Do not permute.

27.7.11.2 AFEU Key Size Register (AFEUKSR)

AFEUKSR — AFEU Key Size Register — Offset 0xC8008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0000															

The value in the AFEU Key Size Register (AFEUKSR) indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in the Key Size Register is ignored. This register is cleared when the AFEU is reset or reinitialized. If the key size specified is less than 1 or greater than 16, a key size error is generated. If the Key Size Register is modified during processing, a context error is generated.

Note: Although the AFEU supports key lengths as short as 1 byte, a 1-byte key offers little security. Most ARC-4 applications specify keys of 5–16 bytes.

The device driver creates properly formatted descriptors for situations requiring a key permute prior to ciphering. When using core processor-controlled access (typically for debug), the user must perform the following sequence:

1. Configure the AFEU Mode Register to perform a permute with key
2. Write the key data to AFEU key registers
3. Write the key size to the Key Size Register.

The AFEU starts permuting the memory with the contents of the key registers immediately after the key size is written.

27.7.11.3 AFEU Context/Data Size Register (AFEUCDSR)

AEFUCDSR AEFU Context Data Size Register Offset 0xC8010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Data Size											
Type	R/W															
Reset	0x0000															

The AFEU Context/Data Size Register (AFEUCDSR) stores the number of bits in the final message, with an upper bound of 4096. Whatever number is written (and whatever truncated value is stored) must be a multiple of 8. This value controls how much data is processed from the last block. The last message block must be a multiple of 8 in the range from 8–64. If a data size that is not a multiple of 8 bits is written, a data size error is generated. Only the least significant 3 bits are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

The AFEUCDSR is also used to specify the context size when context is used. The context size is fixed at 2072 bits (259 bytes). When loading context through the FIFO, all context data must be written prior to writing the context data size. The message data size must be written separately.

Note: When reloading an existing context using core processor-controlled access, the user must write the context to the input FIFO, then write the context size (always 2072 bits). The write of the context size indicates to the AFEU that all context is loaded. The user then writes the message data size to the AFEUCDSR. After this write, the user can begin writing message data to the FIFO.

Writing to this register signals the AFEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated. This register is cleared when the AFEU is reset or reinitialized.

27.7.11.4 AFEU Reset Control Register (AFEURCR)

AFEURCR — AFEU Reset Control Register — Offset 0xC8018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

The AFEU Reset Control Register (AFEURCR) allows 3 levels reset that affect the AFEU only as defined by 3 self-clearing bits. The AFEU executes an internal reset sequence (for hardware reset, SW_RESET, or Module Initialization) that performs proper initialization of the S-Box. To determine when this is complete, observe the AFEUSR[RD] bit (see **Section 27.7.11.5**, *AFEU Status Register (AFEUSR)*, on page 27-267).

Table 27-108. AFEURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupt** Setting this bit causes AFEU interrupts signalling done and error to reset. It further resets the state of the AFEU Interrupt Status Register.	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Module initialization is almost the same as a software reset except that the Interrupt Mask Register remains unchanged.	0 No reset 1 Reset most of AFEU.
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the AFEU. When SR clears, the AFEU enters a routine to perform a proper initialization of the S-Box.	0 No reset 1 Full AFEU reset.

27.7.11.5 AFEU Status Register (AFEUSR)

AFEUSR — AFEU Status Register — Offset 0xC8028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—								OFL							
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IFL								—		HALT	—		EI	DI	RD
Type	R															
Reset	0x0000															

The AFEU Status Register (AFEUSR) reflects the state of AFEU internal signals. The AFEUSR is read-only. Writing to this location results in an address error being reflected in the AFEUISR (see **Section 27.7.11.6**, *AFEU Interrupt Status Register (AFEUISR)*, on page 27-268).

Table 27-109. AFEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–24	0	Reserved. Write to zero for future compatibility.	
OFL 23–16	0	**Output FIFO Length** The number of 8-byte sets currently in the output FIFO.	
IFL 15–8	0	**Input FIFO Length** The number of 8-byte sets currently in the input FIFO.	
— 7–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the AFEU is halted due to an error. **Note:** Because the error causing the AFEU to stop operating can be masked before reaching the Interrupt Status Register, the AFEUISR is used to provide a second source of information regarding errors that prevent normal operation.	0 AFEU not halted 1 AFEU halted
— 4–3	0	Reserved. Write to zero for future compatibility.	
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186).	0 AFEU is not signaling error 1 AFEU is signaling error

Table 27-109. AFEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 AFEU is not signaling done 1 AFEU is signaling done
RD 0	0	**Reset Done** This status bit, when high, indicates that the AFEU has completed its reset sequence. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.11.6 AFEU Interrupt Status Register (AFEUISR)

AFEUISR — AFEU Interrupt Status Register — Offset 0xC8030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—			IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Type	R															
Reset	0x0000															

The AFEU Interrupt Status Register (AFEUISR) indicates the unmasked errors that have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the AFEU Interrupt Mask Register is zero (see **Section 27.6.5.7**, *AFEU Interrupt Mask Register*, on page 27-75).

If the AFEU Interrupt Status Register is non-zero, the AFEU halts and the AFEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186). In addition, if the AFEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the AFEU Reset Control Register. The definition of each bit field in the AFEUISR is listed in **Table 27-110**.

Table 27-110. AFEUISR Field Descriptions

Name	Reset	Description	Settings
— 63–13	0	Reserved. Write to zero for future compatibility.	
IE 12	0	**Internal Error** Indicates whether an internal processing error was detected while the AFEU was performing encryption.	0 No internal error detected. 1 Internal error.
ERE 11	0	**Early Read Error** Indicates whether the AFEU context memory or control was read while the AFEU was performing encryption.	0 No early read error detected. 1 Early read error.
CE 10	0	**Context Error** If set, indicates that AESU key register, the Key Size Register, Data Size Register, Mode Register, or IV register was modified while the AESU was processing.	0 No context error detected. 1 Context error.
KSE 9	0	**Key Size Error** IF set, indicates that a value outside the range 1–16 bytes written to the AFEU Key Size Register.	0 No key size error detected. 1 Key size error.
DSE 8	0	**Data Size Error** If set, indicates that a value not a multiple of 8 bits was written to the AFEU Data Size Register.	0 No data size error detected. 1 Data size error.
ME 7	0	**Mode Error** If set, indicates that an illegal value was detected in the Mode Register, likely caused by writing to a reserved bit in that register.	0 Valid data. 1 Reserved or invalid mode selected.
AE 6	0	**Address Error** If set, an illegal read or write address was detected within the AFEU address space.	0 No address error detected. 1 Address error detected.
OFE 5	0	**Output FIFO Error** If set, the AFEU output FIFO was detected non-empty upon write of AFEU Data Size Register.	0 No output FIFO error detected. 1 Output FIFO non-empty error.
IFE 4	0	**Input FIFO Error** If set, the AFUE input FIFO was detected non-empty upon generation of a done interrupt.	0 No input FIFO error detected. 1 Input FIFO non-empty error.
— 3	0	Reserved. Write to zero for future compatibility.	

Table 27-110. AFEUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
IFO 2	0	**Input FIFO Overflow** If set, the AFEU input FIFO was pushed while full. **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through core processor-controlled access, the AFEU cannot accept FIFO inputs larger than 256 bytes without overflowing.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
OFU 1	0	**Output FIFO Underflow** If set, the AFEU output FIFO was read while empty.	0 No output FIFO underflow error detected. 1 Output FIFO underflow error.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.11.7 AFEU Interrupt Mask Register (AFEUIMR)

AFEUIMR — AFEU Interrupt Mask Register — Offset 0xC8038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—			IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Type	R															
Reset	0x1000															

The AFEU Interrupt Mask Register (AFEUIMR) controls the result of detected errors. For a given error (as defined in **Section 27.6.5.6**, *AFEU Interrupt Status Register*, on page 27-75), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then

upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Table 27-111. AFEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–13	0	Reserved. Write to zero for future compatibility.	
IE 12	0	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 11	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ME 7	0	**Mode Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFE 5	0	**Output FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFE 4	0	**Input FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFU 1	0	**Output FIFO Underflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.11.8 AFEU End_of_Message Register (AFEUEOMR)

AFEUEOMR — AFEU End_of_ Message Register — Offset 0xC8050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The AFEU End_of_Message Register (AFEUEOMR) is used to signal the AFEU that all data to process is written to the input FIFO. This allows the AFEU to do special processing when it reaches the last block of data. Before this register is written, the AFEU does not process the last block of data in its input FIFO. After this register is written, the AFEU continues normal processing on all but the last block of data, then goes on to processes the last block, using the value in the Data Size Register to determine how much of the block to process. The Data Size Register specifies the number of bits to process, which is a multiple of 8 from 8–64. Once processing of the last block is completed, the AFEU signals done interrupt. If the AFEUMR[DC] bit is set, the context is written to the output FIFO following the last 8 message bytes. A read of the AFEUEOMR always returns a zero value.

27.7.11.9 AFEU Context Memory

The S-Box memory consists of 256 bytes of SRAM, each readable and writable as part of a 64-bit set. The S-Box contents should not be written with data unless that data was previously read from the S-Box. Context data should only be written if the AFEUMR[PP] bit is set (see **Section 27.7.11.1**, *AFEU Mode Register (AFEUMR)*, on page 27-263). After context data is written, the context length must be written to the context/data length register (see **Section 27.7.11.3**, *AFEU Context/Data Size Register (AFEUCDSR)*, on page 27-265). Then, message data can be written. If the Context Registers are written during message processing or the AFEUMR[PP] bit is not set, a context error is generated. Reading context data before the module is done generates an error interrupt. Context data can be written and read either via the Context Registers or via the input and output FIFOs. The user specifies the location through the

AFEUMR[CS] bit (see **Section 27.6.5.1**, *AFEU Mode Register*, on page 27-73). See **Section 27.7.11.12**, *AFEU FIFOs*, on page 27-273 for details about AFEU FIFO addressing.

Note: The AFEU Context Memory is in the range defined by offsets 0xC8100–0xC81FF.

27.7.11.10 AFEU Context Memory Pointer Register (AFEUCMPR)

The context memory pointer register holds the internal context pointers that are updated with each byte of message processed. These pointers correspond to the values of I, J, and Sbox[I+1] in the ARC-4 algorithm. If this register is written during message processing, a context error is generated. When performing ARC-4 operations, the user has the option of performing a new S-Box permutation per packet, or unloading the contents of the S-box (context) and reloading this context prior to processing of the next packet. The S-Box contents (256 bytes) plus the three bytes of the context memory pointers are unloaded and reloaded via the AFEU FIFOs. AFEU Context consists of the contents of the S-Box, as well as three counter values, which indicate the next values to be used from the S-Box. Context must be loaded in the same order in which it was unloaded.

Note: The AFEUCMPR is located at offset 0xC8200.

27.7.11.11 AFEU Key Registers (AFEUKR[1–2])

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-Box, in conjunction with the AFEU Key Size Register. AFEU performs permutation starting with the first byte of key register 0, and uses as many bytes from the two key registers as necessary to complete the permutation. Reading either of these memory locations generates an address error interrupt.

Note: The AFEU key registers are located at the following offsets:
AESUKR1 = Offset 0xC8400.
AESUKR2 = Offset 0xC8408.

27.7.11.12 AFEU FIFOs

AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core processor-controlled operation, a write to anywhere in the AFEU FIFO address space enqueues data to the AFEU input FIFO, and a read from anywhere in the AFEU FIFO address space dequeues data from the AFEU output FIFO.

Note: The AFEU FIFOs reside in the range defined by offsets 0xC8800–0xC8FFF. In the special case where context is written through the input FIFO, the first write must be to address offset range 0xC8E00–0xC8E07. Context reads can be from any address in the FIFO address range.

Writes to the input FIFO go first to a staging register and can be written using byte, 4-byte, or 8-byte accesses. When all 8 bytes of the staging register are written, the entire 8 bytes are automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the AFEUEOMR is written.

The output FIFO is readable using byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, that 8 bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflow caused by reading or writing the AFEU FIFOs are reflected in the AFEUISR.

27.7.12 KEU Registers

27.7.12.1 KEU Mode Register (KEUMR)

KEUMR — KEU Mode Register — Offset 0xCE000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—								GSM	CICV	EDGE	PE	INT	—	ALG	
Type	R/W															
Reset	0x0000															

The KEU Mode Register (KEUMR) contains several bits used to program the KEU. The Mode Register is cleared when the KEU is reset or reinitialized. Setting a reserved mode bit generates a data error. Setting both the GSM and EDGE bits to one generates a data error. If the KEU Mode

Register is modified during processing, a context error is generated. **Table 27-112** describes the KEU Mode Register bit fields.

Table 27-112. KEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–8	0	Reserved. Write to zero for future compatibility.	
GSM 7	0	**Select GSM A5/3 Blocks** GSM A5/3 requires that two 114-bit blocks be produced for every 4.615 mS slot. If GSM = 1, the first read of the output FIFO retrieves the first 64-bits of block 1. The second read of the output FIFO retrieves the next 50-bits of block 1 (the remaining bits of this 64 bits are set to zero). The third read of the output Fifo retrieves the first 64-bits of block 2, and a fourth read of the output FIFO retrieves the next 50-bits of block 2 (the remaining bits of this 64 bits are set to zero). If GSM = 0, 228 contiguous bits can be read with successive reads of the output FIFO. In this case, the core application is responsible for handling the A5/3 block formatting. **Note:** If GSM = 1 and EDGE = 1, an error interrupt is generated.	0 GSM A5/3 blocks not selected 1 GSM A5/3 blocks selected
CICV 6	0	**Compare Integrity Check Values** If set, selects integrity check comparison. If the ICVs do not match, an error interrupt is sent to the channel. This field is valid only when the ALG field is set to a function that uses F9.	0 Normal operation; no ICV comparison. 1 After the ICV is computed, compare it to the data in the KEU ICV_In register.
EDGE 5	0	**Select EDGE A5/3 Blocks** EDGE A5/3 requires that two 348-bit blocks be produced for every 4.615 mS slot. If EDGE = 1, the first five reads of the output FIFO retrieve the first 320-bits of block 1. The sixth read of the output FIFO retrieves the final 28-bits of block 1 (the remaining bits of the sixth 64-bit set are set to zero). The next five reads of the output FIFO retrieve the first 320-bits of block 2. The following read of the output FIFO retrieves the final 28-bits of block 2 (the remaining bits of this 64-bit set are set to zero). If EDGE = 0, 696 contiguous bits can be read with successive reads of the output FIFO. In this case the core application is responsible for handling the A5/3 block formatting. **Note:** If EDGE = 1 and GSM = 1, an error interrupt is generated.	0 EDGE A5/3 blocks not selected 1 EDGE A5/3 blocks selected
PE 4	0	**Process End_of_Message** Enables final processing of last message block fir F9 only. **Note:** The PE bit should be set for F9 operations if the 3G frame (or message) is processed as a whole (not split across multiple descriptors). If the frame is processed across multiple descriptors, this bit should only be set on the descriptor performing F9 processing on the final message block.	0 Prevent final block processing (message incomplete) 1 Enable final block processing (message complete)

Table 27-112. KEUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
INT 3	0	**Initialization** Enables initialization for a new message. **Note:** Set this bit for F8 or F9 operations if the 3G frame (or message) is being processed through a single descriptor. If the frame is split across multiple descriptors, this bit should only be set in the descriptor that processes the first block of the message.	0 Prevent Initialization 1 Enable Initialization
— 2	0	Reserved. Write to zero for future compatibility.	
ALG 1–0	0	**Algorithm Selection** Specifies the functions to perform.	00 Perform F8 function only 01 reserved 10 Perform F9 function only 11 reserved

27.7.12.2 KEU Key Size Register (KEUKSR)

KEUKSR — KEU Key Size Register — Offset 0xCE008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0000															

The KEU Key Size Register (KEUKSR) stores the number of bytes in the key. It should be set to 16 bytes. This register is cleared when the KEU is reset or reinitialized. If a key size is specified that does not match the selected algorithm(s), an illegal key size error is generated.

27.7.12.3 KEU Data Size Register (KEUDSR)

KEUDSR — KEU Data Size Register — Offset 0xCE010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Data Size											
Type	R/W															
Reset	0x0000															

The KEU Data Size Register (KEUDSR) stores the number of bits to process in the final 8 message bytes. Because Kasumi allows for bit level granularity for encryption/decryption, there are no illegal data sizes. The proper bit length of the message must be written to notify the KEU of any padding performed by the core. This register is cleared when the KEU is reset or reinitialized. Writing to this register signals the KEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

27.7.12.4 KEU Reset Control Register (KEURCR)

KEURCR — KEU Reset Control Register — Offset 0xCE018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													CLI	RI	SR
Type	R/W															
Reset	0x0000															

The KEU Reset Control Register (KEURCR) allows 3 levels reset of the KEU as defined by the 3 self-clearing bits:

Table 27-113. KEURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
CLI 2	0	**Clear Interrupts** Setting this bit causes KEU interrupts signalling done and error to reset. It further resets the KEU Interrupt Status Register (KEUISR).	0 Normal operation. 1 Clear interrupts and KEUISR.
RI 1	0	**Reinitialization** Module initialization is almost the same as a software reset except that the Interrupt Mask Register remains unchanged. Completion of the reinitialization is indicated by the KEUSR[RD] bit (see **Section 27.7.12.5**, *KEU Status Register (KEUSR),* on page 27-279).	0 Normal operation. 1 Reinitialize KEU.
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the KEU. When the bit clears, the KEU enters a routine that initializes the parameter memories. Completion of the reinitialization is indicated by the KEUSR[RD] bit (see **Section 27.7.12.5**, *KEU Status Register (KEUSR),* on page 27-279).	0 Normal operation. 1 Full KEU reset.

27.7.12.5 KEU Status Register (KEUSR)

KEUSR — KEU Status Register — Offset 0xCE028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—								OFL							
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IFL								—		HALT	ICCR		EI	DI	RD
Type	R															
Reset	0x0000															

The KEU Status Register (KEUSR) is a read-only register that reflects the state of six status outputs. Writing to this location results in an address error being reflected in the KEU Interrupt Status Register.

Table 27-114. KEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–24	0	Reserved. Write to zero for future compatibility.	
OFL 23–16	1	**Output FIFO Length** The number of 8-byte sets currently in the output FIFO.	
IFL 15–8	1	**Input FIFO Length** The number of 8-byte sets currently in the input FIFO.	
— 7–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the KEU is halted due to an error. **Note:** Because the error causing the KEU to stop operating can be masked before reaching the Interrupt Status Register, the KEU Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 KEU not halted 1 KEU halted (must be reset/reinitialized)
ICCR 4–3	0	**Integrity Check Comparison Result** A passed or failed result is generated only if ICV checking is enabled and the selected algorithm is F9.	00 No integrity check performed. 01 Integrity check comparison passed. 10 Integrity check comparison failed. 11 Reserved.

Table 27-114. KEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 KEU is not signaling error 1 KEU is signaling error
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 KEU is not signaling done 1 KEU is signaling done
RD 0	0	**Reset Done** This status bit, when high, indicates that the RNGU has completed its reset sequence. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.12.6 KEU Interrupt Status Register (KEUISR)

KEUISR — KEU Interrupt Status Register — Offset 0xCE030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	DE	AE	OFE	IFE	—	IFO	OFU	—
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the KEU Interrupt Mask Register is zero (see **Section 27.6.6.7**, *KEU Interrupt Mask Register*, on page 27-80).

If the KEU Interrupt Status Register is non-zero, the KEU halts and the KEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186). In addition, if the KEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU

error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the KEURCR[RI] bit (see **Section 27.7.12.4**, *KEU Reset Control Register (KEURCR)*, on page 27-278).

The definition of each bit in the KEU Interrupt Status Register is listed in **Table 27-115**.

Table 27-115. KEUISR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** If set, indicates that an ICV check was performed on an F9 result and that the supplied ICV did not match the value computed by the KEU.	0 No error detected. 1 Integrity check error.
— 13	0	Reserved. Write to zero for future compatibility.	
IE 12	0	**Internal Error** Indicates whether an internal processing error was detected while the KEU was processing. **Note:** This bit is set any time an enabled error condition occurs. It can only be cleared by setting the corresponding bit in the KEUIMR or by resetting the KEU.	0 No internal error detected. 1 Internal error.
ERE 11	0	**Early Read Error** Indicates whether a KEU context or IV register was read while the KEU was processing.	0 No early read error detected. 1 Early read error.
CE 10	0	**Context Error** If set, indicates that KEU key register, the Key Size Register, Data Size Register, Mode Register, or IV register was modified while the KEU was processing.	0 No context error detected. 1 Context error.
KSE 9	0	**Key Size Error** IF set, indicates that an inappropriate value (not 16 or 32) was written to the KEU Key Size Register.	0 No key size error detected. 1 Key size error.
DSE 8	0	**Data Size Error** If set, a value greater than 64 bits was written to the KEU Data Size Register.	0 No data size error detected. 1 Data size error.
DE 7	0	**Data Error** If set, indicates that invalid data was written to a register or that a reserved mode bit was set.	0 Valid data. 1 Reserved or invalid mode selected.
AE 6	0	**Address Error** If set, an illegal read or write address was detected within the KEU address space.	0 No address error detected. 1 Address error detected.
OFE 5	0	**Output FIFO Error** If set, the KEU output FIFO was detected non-empty upon write of KEU Data Size Register.	0 No output FIFO error detected. 1 Output FIFO non-empty error.

Table 27-115. KEUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
IFE 4	0	**Input FIFO Error** If set, the KEU input FIFO was detected non-empty upon generation of a done interrupt.	0 No input FIFO error detected. 1 Input FIFO non-empty error.
— 3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** If set, the KEUU input FIFO was pushed while full.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
OFU 1	0	**Output FIFO Underflow** If set, the KEU output FIFO was read while empty.	0 No output FIFO underflow error detected. 1 Output FIFO underflow error.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.12.7 KEU Interrupt Mask Register (KEUIMR)

KEUIMR — KEU Interrupt Mask Register — Offset 0xCE038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	DE	AE	OFE	IFE	—	IFO	OFU	—
Type	R															
Reset	0x1000															

The KEU Interrupt Mask register controls the result of detected errors. For a given error (as defined in **Section 27.6.6.6**, *KEU Interrupt Status Register*, on page 27-79), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the KEU Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the KEU Interrupt Status Register is updated to reflect the error, causing

assertion of the error interrupt signal, and causing the module to halt processing. The definition of each bit in the KEU Interrupt Mask Register is listed in **Table 27-116**.

Table 27-116. KEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 13	0	Reserved. Write to zero for future compatibility.	
IE 12	1	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 11	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ME 7	0	**Mode Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFE 5	0	**Output FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFE 4	0	**Input FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFU 1	0	**Output FIFO Underflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.12.8 KEU Data Out Register (KEUDOR) for F9 MAC

KEUDOR — KEU Data Out Register — Offset 0xCE048

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	KEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	KEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	KEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Following a done interrupt, the read-only KEUDOR holds the F9 message authentication code. A 64-bit value is returned. This value can be truncated to 32 bits for some applications. Writing to this location results in an address error being reflected in the KEUISR.

Note: According to the ETSI/SAGE 3GPP specification for F9 (version 1.2), only 32 bits of the final MAC are used. This is the lower 4 bytes of the KEUDOR.

27.7.12.9 KEU End_of_Message Register (KEUEOMR)

KEUEOMR — KEU End_of_ Message Register — Offset 0xCE050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The KEU End_of_Message Register (KEUEOMR) is used to signal to the KEU that the final message block is written to the input FIFO. Writing to this register causes the KEU to process the final block of a message, allowing it to signal a done interrupt. When processing the last block, the value in the Data Size Register determines how many bits of the final message word (1–64) are processed. The value written to this register has no significance. A read of this register always returns a zero value.

27.7.12.10 KEU IV1 Register (KEUIV1R)

KEUIV1R — KEU IV1 Register — Offset 0xCE100

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	CC															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CC															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CB					CD	—		CA							
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CE															
Type	R/W															
Reset	0x0000															

KEU IV1R is a general-purpose IV register used during the initialization phase of F8 algorithms for 3GPP, GSM A5/3, EDGE A5/3, and GPRS GEA3 and for the F9 algorithm for 3GPP. The appropriate value, as defined by the standards for each algorithm, must be written before a new message is started. Once the initialization phase is completed, the KEU IV1 register is not used for the remainder of F8 or F9 processing. However, if 3GPP F9 is selected, the KEU IV_1 register MUST be written back during context switches to complete the generation of the 3GPP MAC, because the KEU IV_1 register contains the Direction bit as defined by the 3GPP standard.

Table 27-117. KEUIV1R Field Descriptions

Name	Reset	Description	Settings for Protocol			
			3GPP	GSM-A5/3	EDGE-A5/3	GPRS-GEA3
CE 63–48	0	**CE Value** Specifies the CE variable value.	0x0000	0x0000	0x0000	0x0000
CA 47–41	0	**CA Value** Specifies the CA variable value.	00000000	00001111	11110000	11111111
— 40–39	0	Reserved. Write to zero for future compatibility.				
CD 38	0	**CD Value** Specifies the CD variable value.	Direction bit	0	0	0
CB 37–32	0	**CB Value** Specifies the CB variable value.	Bearer	00000	00000	00000
CC 31–0	0	**CC Value** Specifies the CC variable value.	Count	Count	0000000000|| Count	Frame-dependent value (32 bits)

Note: The user must ensure that fields of the KEUIV1 register are programmed correctly in accordance with the selected algorithm.

27.7.12.11 KEU ICV_In Register (KEUICVIR)

KEUICVIR — KEU ICV_In Register — Offset 0xCE108

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	ICV_In															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	ICV_In															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	ICV_In															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ICV_In															
Type	R															
Reset	0x0000															

If ICV checking is required, the KEUICVIR holds the value to be compared with the computed F9 MAC value. The value must be written to KEUICVIR before the data size is written.

27.7.12.12 KEU IV2 Register (KEUIV2R)

KEUIV2R — KEU IV2 Register — Offset 0xCE110

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	FRESH															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	FRESH															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	R/W															
Reset	0x0000															

The KEUIV2R hold the FRESH value that is used during the initialization phase of the 3GPP F9 algorithm. This value is ignored when the F8 algorithm is selected. The FRESH value must be written before starting a new message to be processed with 3GPP F9. Once the initialization phase is completed, KEUIV2R is not used during message processing. KEUIV2R does not need to be written during context switches. FRESH is a 32-bit value that occupies the upper half of the register.

27.7.12.13 KEU Context 1–6 Registers (KEUCR[1–6])

KEUCR1 KEU Context Registers Offset 0xCE118
KEUCR2 Offset 0xCE120
KEUCR3 Offset 0xCE128
KEUCR4 Offset 0xCE130
KEUCR5 Offset 0xCE138
KEUCR6 Offset 0xCE140

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	Context															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	Context															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Context															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Context															
Type	R/W															
Reset	0x0000															

There are six 64-bit KEU context data registers that allow the core to read/write the contents of the context used to process the message. The KEUCDRs must be read when changing context and restored to their original values to resume processing an interrupted message. For F8 and 3GPP F9 modes, all 6 64-bit KEUCDRs must be read to retrieve context, and all 6 must be written back to restore context. The context must be written prior to the key data. If the any of the KEU context data registers are written during message processing, a context error is generated. All KEUCDRs are cleared when a hard/soft reset or initialization is performed.

Note: In typical operation, a frame is received and processed in its entirety, with the KEU performing session specific initialization using the contexts of KEU IV_1 and IV_2 registers. The KEU Context Data and IV_1 registers should only be unloaded/reloaded when the processing of a frame is discontinued prior to completion, then processing is resumed.

27.7.12.14 KEU Key Data Registers 1–2 (KEUKDR[1–2])

KEUKDR1 KEU Key Data Registers 1–2 Offset 0xCE400
KEUKDR2 Offset 0xCE408

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	CK															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CK															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CK															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CK															
Type	W															
Reset	0x0000															

The first two KEU Key Data Registers together hold one 128-bit key that is used for F8 encryption/decryption. KEUKDR1 (CK-high) holds the first 8 bytes (1–8). KEUKDR2 (CK-low) holds the second 8 bytes (9–16). The KEU Key Data Registers must be written before message processing begins and cannot be written while the block is processing data, or a context error occurs. Reading from either of these registers results in an address error being reflected in the KEUISR.

27.7.12.15 KEU Key Data Registers 3–4 (KEUKDR[3–4])

KEUKDR3 KEU Key Data Registers 3–4 Offset 0xCE410
KEUKDR4 Offset 0xCE418

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	IK															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	IK															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	IK															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IK															
Type	W															
Reset	0x0000															

The KEUKDR[3–4] together hold one 128-bit key used for F9 message authentication. KEUKDR3 (IK-high) holds the first 8 bytes (1–8). KEUKDR4 (IK-low) holds the second 8 bytes (9–16). The KEU Key Data Registers must be written before message processing begins and cannot be written while the block is processing data, or a context error occurs.

If the ‘F9 only’ mode is set, the integrity key data can be optionally written to KEUKDR[1–2]. This eliminates the need for the core to offset from the base key address to write to KEUKDR[3–4] while using the KEU exclusively for the F9 integrity function. Reading from either of these registers results in an address error being reflected in the KEUISR.

27.7.12.16 KEU Input FIFO/Output FIFO

The KEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For core-controlled operation, a write to anywhere in the KEU FIFO address space enqueues data to the KEU input FIFO, and a read from anywhere in the KEU FIFO address space dequeues data from the KEU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the staging register are written, the entire 8-bytes is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the KEU End_of_Message Register is written.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When all 8 bytes of the header are read, that 8 bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflows caused by reading or writing the KEU FIFOs are reflected in the KEU Interrupt Status Register.

The KEU fetches data 64 bits at a time from the KEU Input FIFO. During F8 processing, the input data is XORed with the generated keystream and the results are placed in the KEU Output FIFO. During F9 processing, the input data is hashed with the integrity key and the resulting MAC is placed in the KEU data out register. The output size is the same as the input size.

Note: The KEU input FIFO and output FIFO are located at offset 0xCE800–0xCEFFF.

27.7.13 CRCU Registers

27.7.13.1 CRCU Mode Register (CRCUMR)

CRCUMR — CRCU Mode Register — Offset 0xCF000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—									CICV	—	DOC	DOS	DIS	ALG	
Type	R/W															
Reset	0x0000															

The CRCU Mode Register (CRCUMR) contains several bits used to program the CRCU and is generally the first register written. The Mode Register is cleared when the CRCU is reset or reinitialized. Setting a reserved mode bit generates a data error. If the CRCU Mode Register is

modified during processing, a context error is generated. **Table 27-118** describes the CRCU Mode Register bit fields.

Table 27-118. CRCUMR Field Descriptions

Name	Reset	Description	Settings
— 63–7	0	Reserved. Write to zero for future compatibility.	
CICV 6	0	**Compare Integrity Check Values** If set, selects integrity check comparison, which compares the CRC calculated across the message and received ICV against the stored residue. The comparison is performed prior to the bit manipulations controlled by the DOS and DOC bits.	0 Comparison disabled. 1 Enable ICV check.
— 5	0	Reserved. Write to zero for future compatibility.	
DOC 4	0	**Disable Output Complement** The normal processing includes complementing the CRC result.	0 Normal operation. CRC result is complemented. 1 Disable output complement.
DOS 3	0	**Disable Output Swap** The normal processing includes a bit swap and byte swap of the result CRC.	0 Normal operation. CRC result is bit-swapped and byte-swapped. 1 Disable output swapping.
DIS 2	0	**Disable Input Swap** The normal processing includes a bit swap and byte swap of the input data.	0 Normal operation. Input data is bit-swapped and byte-swapped. 1 Disable input swapping.
ALG 1–0	0	**Algorithm Selection** Specifies the CRC algorithm for the CRCU.	00 **IEEE** Std. 802 mode. The CRC32 algorithm is performed using the polynomial 0x04C11DB7 and the residue 0xC704DD7B is used for ICV checking. 01 iSCSI mode. The CRC32 algorithm is performed using the polynomial 0x1EDC6F41 and the residue 0x1C29D19ED is used for ICV checking. 10 Static custom mode. The CRC remainder is computed using the Control Register bits 31–0 as the polynomial and the bits 63–32 as the residue. 11 Dynamic custom mode. The CRC remainder is computed using the Key Register bits 31–0 as the polynomial and bits 63–32 as the residue.

27.7.13.2 CRCU Key Size Register (CRCUKSR)

CRCUKSR — CRCU Key Size Register — Offset 0xCF008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0000															

The CRCU Key Size Register (CRCUKSR) stores the number of bytes in the dynamic polynomial written to the CRCU Key Register (see **Section 27.7.13.12**). This register is cleared when the CRCU is reset or reinitialized. If a key size other than zero, four, or eight is written to it, an illegal key size error is generated. Because the polynomial size is clearly defined by the selected algorithm, it is not necessary to write to this register. A context error is generated if this register is written after processing begins.

27.7.13.3 CRCU Data Size Register (CRCUDSR)

CRCUDSR — CRCU Data Size Register — Offset 0xCF010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Data Size											
Type	R/W															
Reset	0x0000															

The CRCU Data Size Register (CRCUDSR) stores the number of bits to process. Writing to this register put the CRCU into a busy state and starts data processing. The register can be written multiple times during data processing. The actual values are ignored, but an error is generated if the value is not a multiple of 8 bits.

27.7.13.4 CRCU Reset Control Register (CRCURCR)

CRCURCR — CRCU Reset Control Register — Offset 0xCF018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

The CRCU Reset Control Register (CRCURCR) controls the reset/reinitialization of the block:

Table 27-119. CRCURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupts** Setting this bit causes CRCU interrupts signalling done and error to reset. It further resets the CRCU Interrupt Status Register (CRCUISR).	0 Do not reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Module initialization is almost the same as a software reset except that the CRCU Interrupt Mask Register remains unchanged. Completion of the initialization is indicated by the CRCUSR[RD] bit (see **Section 27.7.13.6**, *CRCU Status Register (CRCUSR),* on page 27-298).	0 Do not initialize. 1 Initialize CRCU.
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the CRCU. When the bit clears, the CRCU enters a routine that initializes the parameter memories. Completion of the initialization is indicated by the CRCUSR[RD] bit (see **Section 27.7.13.6**, *CRCU Status Register (CRCUSR),* on page 27-298).	0 Do not reset. 1 Full CRCU reset.

27.7.13.5 CRCU Control Register (CRCUCR)

CRCUCR — CRCU Control Register — Offset 0xCF020

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	r^{31}	r^{30}	r^{29}	r^{28}	r^{27}	r^{26}	r^{25}	r^{24}	r^{23}	r^{22}	r^{21}	r^{20}	r^{19}	r^{18}	r^{17}	r^{16}
Type	R/W															
Reset	0xC704															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	r^{15}	r^{14}	r^{13}	r^{12}	r^{11}	r^{10}	r^{9}	r^{8}	r^{7}	r^{6}	r^{5}	r^{4}	r^{3}	r^{2}	r^{1}	r^{0}
Type	R/W															
Reset	0xDD7B															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	g^{31}	g^{30}	g^{29}	g^{28}	g^{27}	g^{26}	g^{25}	g^{24}	g^{23}	g^{22}	g^{21}	g^{20}	g^{19}	g^{18}	g^{17}	g^{16}
Type	R/W															
Reset	0x04C1															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	g^{15}	g^{14}	g^{13}	g^{12}	g^{11}	g^{10}	g^{9}	g^{8}	g^{7}	g^{6}	g^{5}	g^{4}	g^{3}	g^{2}	g^{1}	g^{0}
Type	R/W															
Reset	0x1DB7															

The Control Register stores the static polynomial and residue used in custom CRC computations. This register is static, in that it is reset by performing a software reset, not by an EU reinitialize. This allows a platform-specific custom polynomial to be written to the register once and used many times. A context error is generated if this register is written after processing begins. A polynomial error is generated if a value is written to this register which does not have a one in bit g^0.

Note: The default reset value is the **IEEE** 802 standard CRC32 residue coefficient r and polynomial coefficient g.

Table 27-120 indicates the bit position of each of the coefficients.

Table 27-120. CRCUCR Field Descriptions

Name	Reset	Description
r^{31}–r^0 63–32	1	**Residue Coefficient Bit Positions 31–0**
g^{31}–g^0 31–0	1	**Polynomial Coefficient Bit Positions 31–0** **Note:** The value of g^0 must be 1.

27.7.13.6 CRCU Status Register (CRCUSR)

CRCUSR — CRCU Status Register — Offset 0xCF028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IFL								—		HALT	ICCR		EI	DI	RD
Type	R															
Reset	0x0000															

The CRCU Status Register (CRCUSR) is a read-only register that provides general information about the status of the CRCU. Writing to this location is ignored.

Table 27-121. CRCUSR Field Descriptions

Name	Reset	Description	Settings
— 63–16	0	Reserved. Write to zero for future compatibility.	
IFL 15–8	1	**Input FIFO Length** The number of 8-byte sets currently in the input FIFO.	
— 7–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the CRCU is halted due to an error. **Note:** Because the error causing the CRCU to stop operating can be masked before reaching the Interrupt Status Register, the CRCU Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 CRCU not halted 1 CRCU halted (must be reset/reinitialized)
ICCR 4–3	0	**Integrity Check Comparison Result** A passed or failed result is generated only if ICV checking is enabled and the selected algorithm is F9.	00 No integrity check performed. 01 Integrity check comparison passed. 10 Integrity check comparison failed. 11 Reserved.
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 CRCU is not signaling error 1 CRCU is signaling error

Table 27-121. CRCUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 CRCU is not signaling done 1 CRCU is signaling done
RD 0	0	**Reset Done** This status bit, when high, indicates that the RNGU has completed its reset sequence. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.13.7 CRCU Interrupt/Error Status Register (CRCUISR)

CRCUISR — CRCU Interrupt/Error Status Register — Offset 0xCF030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	PE	IE	ERE	CE	KSE	DSE	DE	AE	—			IFO	—	
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the CRCU Interrupt Mask Register is zero (see **Section 27.7.13.8**, *CRCU Interrupt/Error Mask Register (CRCUIMR)*, on page 27-301).

If the CRCU Interrupt Status Register is non-zero, the CRCU halts and the CRCU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186). In addition, if the CRCU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the core, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. All bits in this register except the IE bit can also be cleared by setting the CRCURCR[RI] bit (see **Section 27.7.13.4**, *CRCU Reset Control Register (CRCURCR)*, on page 27-296). The IE bit can only be cleared by resetting/initializing the CRCU.

The definition of each bit in the CRCU Interrupt Status Register is listed in **Table 27-122**.

Table 27-122. CRCUISR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** If set, indicates that an ICV check was performed on an F9 result and that the supplied ICV did not match the value computed by the CRCU.	0 No error detected. 1 Integrity check error.
PE 13	0	**Polynomial Error** If set, indicates that an invalid polynomial was written to the CRCU Key Register or the CRCU Control Register.	0 No error detected. 1 Invalid polynomial error.
IE 12	0	**Internal Error** Indicates whether an internal processing error was detected while the CRCU was processing. **Note:** This bit is set any time an enabled error condition occurs. It can only be cleared by setting the corresponding bit in the CRCUIMR or by resetting the CRCU.	0 No internal error detected. 1 Internal error.
ERE 11	0	**Early Read Error** Indicates whether a CRCU context or IV register was read while the CRCU was processing.	0 No early read error detected. 1 Early read error.
CE 10	0	**Context Error** If set, indicates that the CRCU key register, the key size register, data size register, mode register, or IV register was modified while the CRCU was processing.	0 No context error detected. 1 Context error.
KSE 9	0	**Key Size Error** If set, indicates that an inappropriate value (not 16 or 32) was written to the CRCU Key Size Register.	0 No key size error detected. 1 Key size error.
DSE 8	0	**Data Size Error** If set, indicates one of the following three conditions: • A value that was not a multiple of 8 bits was written to the CRCU data size register • Data was written to the CRCU end-of-message register before writing to the CRCU data size register. • Data was written to the CRCU input FIFO before writing to the CRCU data size register.	0 No data size error detected. 1 Data size error.

Table 27-122. CRCUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
DE 7	0	**Data Error** If set, indicates that invalid data was written to a register or that a reserved mode bit was set.	0 Valid data. 1 Reserved or invalid mode selected.
AE 6	0	**Address Error** If set, an illegal read or write address was detected within the CRCU address space.	0 No address error detected. 1 Address error detected.
— 5–3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** If set, the CRCUU input FIFO was pushed while full.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
— 1–0	0	Reserved. Write to zero for future compatibility.	

27.7.13.8 CRCU Interrupt/Error Mask Register (CRCUIMR)

CRCUIMR — CRCU Interrupt/Error Mask Register — Offset 0xCF038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	PE	IE	ERE	CE	KSE	DSE	DE	AE	—			IFO	—	
Type	R															
Reset	0x1000															

The CRCU Interrupt Mask register controls the result of detected errors. For a given error (as defined in **Section 27.7.13.7**, *CRCU Interrupt/Error Status Register (CRCUISR)*, on page 27-299), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the CRCU Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the CRCU Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the

module to halt processing. The definition of each bit in the CRCU Interrupt Mask Register is listed in **Table 27-123**.

Table 27-123. CRCUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
PE 13	1	**Polynomial Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IE 12	1	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 11	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DE 7	0	**Data Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 5–3	0	Reserved. Write to zero for future compatibility.	
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 1–0	0	Reserved. Write to zero for future compatibility.	

27.7.13.9 CRCU ICV Size Register (CRCUICVSR)

CRCUICVSR CRCU ICV Size Register Offset 0xCF040

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	R/W															
Reset	0x0000															

The CRCU ICV size register is 64-bit readable and writable for compatibility with the MDEU. Values written to this location are always ignored and a read from this location always returns 0. A context error is generated if this register is written after processing begins.

27.7.13.10 CRCU End_of_Message Register (CRCUEOMR)

CRCUEOMR CRCU End_of_ Message Register Offset 0xCF050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The CRCU End_of_Message Register (CRCUEOMR) is used to signal to the CRCU that the final message block is written to the CRCU. A write to this register is required to complete a CRC32 operation. The CRCU starts processing message data as soon as the CRCU data size register is written and data becomes available in the input FIFO, but it does not process a remaining partial word or perform an ICV check until a write to this register occurs. The written value is not used. Reading this register always returns 0.

27.7.13.11 CRCU Context Register (CRCUCXR)

CRCUCXR — CRCU Context Register — Offset 0xCF100

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	CRCR															
Type	R/W															
Reset	0xffff															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CRCRr															
Type	R/W															
Reset	0xffff															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	R/W															
Reset	0x0000															

The CRCU context register contains the CRC result when processing is complete. The result is stored in the upper half of the register and the lower half is not used. The register can be written with an intermediate CRC result or desired initial state prior to processing any data. The reset state for the processing field is all 1s because this allows the CRC32 algorithm to detect bit errors in the leading zeros of a message. A context error is generated if this register is written after processing begins. An early read error is generated if this register is read while the module is busy. Actual manipulation of the data is determined by settings in the CRCU Mode Register (see **Section 27.7.13.1**)

Table 27-124. CRCUCXR Field Descriptions

Name	Reset	Description
CRCR 63–32	0xFFFFFFFF	**CRC Residue**
— 31–0	0	Reserved. Write to zero for future compatibility.

27.7.13.12 CRCU Key Register (CRCUKR)

CRCUKR — CRCU Key Register — Offset 0xCF400

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	r^{31}	r^{30}	r^{29}	r^{28}	r^{27}	r^{26}	r^{25}	r^{24}	r^{23}	r^{22}	r^{21}	r^{20}	r^{19}	r^{18}	r^{17}	r^{16}
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	r^{15}	r^{14}	r^{13}	r^{12}	r^{11}	r^{10}	r^{9}	r^{8}	r^{7}	r^{6}	r^{5}	r^{4}	r^{3}	r^{2}	r^{1}	r^{0}
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	g^{31}	g^{30}	g^{29}	g^{28}	g^{27}	g^{26}	g^{25}	g^{24}	g^{23}	g^{22}	g^{21}	g^{20}	g^{19}	g^{18}	g^{17}	g^{16}
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	g^{15}	g^{14}	g^{13}	g^{12}	g^{11}	g^{10}	g^{9}	g^{8}	g^{7}	g^{6}	g^{5}	g^{4}	g^{3}	g^{2}	g^{1}	g^{0}
Type	R/W															
Reset	0x0001															

The Key Register stores the polynomial and residue for the dynamic custom mode if selected by the CRCU Mode Register (see **Section 27.7.13.1**). This register is dynamic, in that it is reset by performing a reinitialize or a software reset. This allows a custom polynomial to be used for specific processing without changing the platform-specific static custom polynomial stored in the CRC Control Register (see **Section 27.7.13.5**). A residue does not need to be programmed unless ICV checking is being performed. A context error is generated if this register is written after processing begins. A polynomial error is generated if a value is written to this register which does not have a one in bit g^0. **Table 27-125** indicates the bit position of each of the coefficients.

Table 27-125. CRCUKR Field Descriptions

Name	Reset	Description
r^{31}–r^0 63–32	0	**Residue Bit Positions 31–0**
g^{31}–g^0 31–0	1	**Polynomial Bit Positions 31–0** **Note:** The value of g^0 must be 1.

27.7.13.13 CRCU Input FIFO

Words written to this address range are pushed onto the CRCU input FIFO, thereby buffering them for processing. Partial words and misaligned data can be written to this address and it is automatically realigned based on a big endian byte order.

The CRCU input FIFO is located at offset 0xCF800–0xCFFFF.

27.7.14 STEU Registers

27.7.14.1 STEU Mode Register (STEUMR)

STEUMR — STEU Mode Register — Offset 0xCD000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—									CICV	—	PE	INT	ALG		
Type	R/W															
Reset	0x0000															

The STEU Mode Register (STEUMR) contains several bits used to program the STEU. The Mode Register is cleared when the STEU is reset or reinitialized. Selecting an undefined mode generates an illegal mode error. If the STEU Mode Register is modified during processing, a context error is generated. **Table 27-112** describes the STEU Mode Register bit fields.

Table 27-126. STEUMR Field Descriptions

Name	Reset	Description	Settings
— 63–7	0	Reserved. Write to zero for future compatibility.	
CICV 6	0	**Compare Integrity Check Values** If set, selects integrity check comparison. If the ICVs do not match, an error interrupt is sent to the channel. This field is valid only when the ALG field is set to a function that uses F9.	0 Normal operation; no ICV comparison. 1 After the ICV is computed, compare it to the data in the STEU ICV_In register.
— 5	0	Reserved. Write to zero for future compatibility.	
PE 4	0	**Process End_of_Message** Enables final processing of last message block for F9 only. **Note:** The PE bit should be set for F9 operations if the 3G frame (or message) is processed as a whole (not split across multiple descriptors). If the frame is processed across multiple descriptors, this bit should only be set on the descriptor performing F9 processing on the final message block.	0 Prevent final block processing (message incomplete) 1 Enable final block processing (message complete)

Table 27-126. STEUMR Field Descriptions (Continued)

Name	Reset	Description	Settings
INT 3	0	**Initialization** Enables initialization for a new message. **Note:** Set this bit for F8 or F9 operations if the 3G frame (or message) is being processed through a single descriptor. If the frame is split across multiple descriptors, this bit should only be set in the descriptor that processes the first block of the message.	0 Prevent Initialization 1 Enable Initialization
ALG 2–0	0	**Algorithm Selection** Specifies the functions to perform. Entry of any of the reserved modes generates an illegal mode error.	001 Perform F8 function only 010 Perform F9 function only All other values are reserved.

27.7.14.2 STEU Key Size Register (STEUKSR)

STEUKSR STEU Key Size Register Offset 0xCD008

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—				Key Size											
Type	R/W															
Reset	0x0010															

The STEU Key Size Register (STEUKSR) does not physically exist because the key size is always 16 bytes for the SNOW3G algorithms. However, if an illegal key size is written to this address, an illegal key size error is generated.

27.7.14.3 STEU Data Size Register (STEUDSR)

STEUDSR STEU Data Size Register Offset 0xCD010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—												Data Size in bits			
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Data Size in bits															
Type	R/W															
Reset	0x0000															

The STEU Data Size Register (STEUDSR) stores the number of bits to process. This value must be written prior to loading message data and after the STEU mode register is written. The STEU decrements the value in this register as it processes the message. The STEUDSR can be read at any time to determine the number of bits remaining to be processed. The STEU continues to process data until the value in the STEUDSR reaches 0. For F9 mode, the data size must be divisible by 64 when the STEUMR[PE] bit is set (see **Section 27.7.14.1**). Therefore, for multi-session processing, the message can only be split on 64-bit boundaries. Violating this rule generates an illegal data size error. If a 0 is written to the STEUDSR followed by a write to the STEUEOMR (see **Section 27.7.14.9**) or if the write to STEUEOMR occurs without writing the the STEUDSR, the STEU asserts a done interrupt signalling that processing is complete.

27.7.14.4 STEU Reset Control Register (STEURCR)

STEURCR — STEU Reset Control Register — Offset 0xCD018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													CLI	RI	SR
Type	R/W															
Reset	0x0000															

The STEU Reset Control Register (STEURCR) determines the type of STEU reset as defined by the 3 self-clearing bits: Reading this register returns a 0.

Table 27-127. STEURCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
CLI 2	0	**Clear Interrupts** Setting this bit causes STEU interrupts clears the errors in the STEU Error Status Register and deasserts the error and done interrupt signals. The reset done signal is deasserted for one cycle following a CLI reset. This bit is self-clearing and also clears the EOM register and the ICV Error/Pass bits of the Status Register.	0 Do not clear interrupts/errors. 1 Clear interrupts/ errors.
RI 1	0	**Reinitialization** Setting this bit clears all registers except the Error Status Register. The bit is self-clearing and asserts the reset done interrupt when initialization is complete. Completion of the reinitialization is indicated by the STEUSR[RD] bit (see **Section 27.7.14.5**, *STEU Status Register (STEUSR),* on page 27-311).	0 No initialization. 1 Initialize STEU.
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the STEU. The bit is self-clearing and asserts the reset done interrupt when initialization is complete. Completion of the reinitialization is indicated by the STEUSR[RD] bit (see **Section 27.7.14.5**, *STEU Status Register (STEUSR),* on page 27-311).	0 Normal operation. 1 Full STEU reset.

27.7.14.5 STEU Status Register (STEUSR)

STEUSR — STEU Status Register — Offset 0xCD028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—								OFL							
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	IFL								—		HALT	ICCR		EI	DI	RD
Type	R															
Reset	0x0000															

The STEU Status Register (STEUSR) is a read-only register that reflects the current state of the STEU.

Table 27-128. STEUSR Field Descriptions

Name	Reset	Description	Settings
— 63–24	0	Reserved. Write to zero for future compatibility.	
OFL 23–16	1	**Output FIFO Length** The number of 8-byte sets currently in the output FIFO.	
IFL 15–8	1	**Input FIFO Length** The number of 8-byte sets currently in the input FIFO.	
— 7–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the STEU an/or the F9 mode state machine are halted due to an internal error. or an error condition flagged in the STEU Error Status Register. A reset is required for recovery.	0 Engine not halted 1 Engine halted (must be reset)
ICCR 4–3	0	**Integrity Check Comparison Result** A passed or failed result is generated only if ICV checking is enabled and the selected algorithm is F9.	00 No integrity check performed. 01 Integrity check comparison passed. 10 Integrity check comparison failed. 11 Reserved.
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 STEU is not signaling error 1 STEU is signaling error

Table 27-128. STEUSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit reflects the state of the done interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 STEU is not signaling done 1 STEU is signaling done
RD 0	0	**Reset Done** This status bit, when high, indicates that the STEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel.	0 Reset in progress. 1 Reset done.

27.7.14.6 STEU Interrupt Status Register (STEUISR)

STEUISR — STEU Interrupt Status Register — Offset 0xCD030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	IM	IE	ERE	CE	KSE	DSE	—	AE	OFE	IFE	IFU	IFO	OFU	—
Type	R															
Reset	0x0000															

The STEU Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the STEU Interrupt Mask Register is zero (see **Section 27.7.14.7**, *STEU Interrupt Mask Register (STEUIMR)*, on page 27-314).

If the STEU Interrupt Status Register is non-zero, the STEU halts and the STEU error interrupt signal is asserted to the controller (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186). In addition, if the STEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller.

To recover and start new processing after an error, the STEU must be reset (using the STEURCR[SR], see **Section 27.7.14.4**).

The definition of each bit in the STEU Interrupt Status Register is listed in **Table 27-115**.

Table 27-129. STEUISR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** If set, indicates that an ICV check was performed on an F9 result and that the supplied ICV did not match the value computed by the STEU.	0 No error detected. 1 Integrity check error.
IM 13	0	**Illegal Mode Error** If set, indicates that an invalid value was written to the MODE field of the STEU Mode Register.	0 No error detected. 1 Illegal mode error.
IE 12	0	**Internal Error** Indicates whether an internal processing error was detected while the STEU was processing. **Note:** This bit is set any time an enabled error condition occurs. It can only be cleared by resetting the STEU.	0 No internal error detected. 1 Internal error.
ERE 11	0	**Early Read Error** Indicates whether a STEU context or LFSR/FSM State Register was read while the STEU was processing.	0 No early read error detected. 1 Early read error.
CE 10	0	**Context Error** If set, indicates that STEU Mode Register, Key Data Register, Data Size Register, or a Context Register was modified while the STEU was processing.	0 No context error detected. 1 Context error.
KSE 9	0	**Key Size Error** IF set, indicates that an illegal key size was written to the STEU Key Size Register.	0 No key size error detected. 1 Key size error.
DSE 8	0	**Data Size Error** If set, F9 mode selected and the PE bit in the Mode register is 0 and the data size is not a multiple of 64	0 No data size error detected. 1 Data size error.
— 7	0	Reserved. Write to zero for future compatibility.	
AE 6	0	**Address Error** If set, an illegal read or write address was detected within the STEU address space. **Note:** This error results from an access to an undefined address, or a write to the Status Register, or an illegal byte enable configuration for the accessed address. All STEU registers below offset 0x100 must be accessed as a whole (8 bytes) or in halves (lower 4 bytes or upper 4 bytes). All STEU registers above and including offset 0x100 can be accessed with any combination of byte enables.	0 No address error detected. 1 Address error detected.
OFE 5	0	**Output FIFO Error** If set, the STEU output FIFO was detected non-empty upon write of STEU Data Size Register.	0 No output FIFO error detected. 1 Output FIFO non-empty error.

Table 27-129. STEUISR Field Descriptions (Continued)

Name	Reset	Description	Settings
IFE 4	0	**Input FIFO Error** If set, the STEU input FIFO was detected non-empty upon generation of a done interrupt.	0 No input FIFO error detected. 1 Input FIFO non-empty error.
IFU 3	0	**EOM Before Input FIFO Write End Error** If set, the EOM was issued before all data is written to the STEU input FIFO.	0 No error detected. 1 Not all data written to the input FIFO.
IFO 2	0	**Input FIFO Overflow** If set, the STEU input FIFO was pushed while full.	0 No input FIFO overflow error detected. 1 Input FIFO overflow error.
OFU 1	0	**Output FIFO Underflow** If set, the STEU output FIFO was read while empty.	0 No output FIFO underflow error detected. 1 Output FIFO underflow error.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.14.7 STEU Interrupt Mask Register (STEUIMR)

STEUIMR — STEU Interrupt Mask Register — Offset 0xCD038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	ICE	IM	IE	ERE	CE	KSE	DSE	—	AE	OFE	IFE	IFU	IFO	OFU	—
Type	R															
Reset	0x1000															

The STEU Interrupt Mask register controls the result of detected errors. For a given error (as defined in **Section 27.7.14.6**, *STEU Interrupt Status Register (STEUISR)*, on page 27-312), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the STEU Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the STEU Interrupt Status Register is updated to reflect the

error, causing assertion of the error interrupt signal, and causing the module to halt processing. The definition of each bit in the STEU Interrupt Mask Register is listed in **Table 27-116**.

Table 27-130. STEUIMR Field Descriptions

Name	Reset	Description	Settings
— 63–15	0	Reserved. Write to zero for future compatibility.	
ICE 14	0	**Integrity Check Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IM 13	0	**Illegal Mode Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IE 12	1	**Internal Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
ERE 11	0	**Early Read Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
CE 10	0	**Context Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
KSE 9	0	**Key Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
DSE 8	0	**Data Size Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 7	0	Reserved. Write to zero for future compatibility.	
AE 6	0	**Address Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFE 5	0	**Output FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFE 4	0	**Input FIFO Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFU 3	0	**EOM Before Input FIFO Write End Error** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
IFO 2	0	**Input FIFO Overflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
OFU 1	0	**Output FIFO Underflow** Enables/disables interrupt generation.	0 Interrupt enabled. 1 Interrupt disabled.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.14.8 STEU Data Out Register (STEUDOR) for F9 MAC

STEUDOR — STEU Data Out Register — Offset 0xCD048

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	STEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	STEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	STEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	STEU Data Out (F9 MAC)															
Type	R															
Reset	0x0000															

STEUDOR is physically the same as the STEU Context Register 1, except that the STEUDOR can be read at any time, even during processing, while the context register can only be read after processing is complete. A write to the STEUDOR is ignored. Reading STEUDOR returns the contents of STEU Context Register 1 which is used to compute the MAC produced by F9 mode. Following a done interrupt, the read-only STEUDOR holds the F9 message authentication code. A 64-bit value is returned. If this register is read after processing is complete, the upper 32 bits are always equal to 0.

27.7.14.9 STEU End_of_Message Register (STEUEOMR)

STEUEOMR — STEU End_of_ Message Register — Offset 0xCD050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The STEU End_of_Message Register (STEUEOMR) is used to signal to the STEU that the final message block is written to the input FIFO. Writing to this register causes the STEU to process the remaining data in the input FIFO and generate a done interrupt. The value written to this register has no significance. A read of this register always returns a zero value.

27.7.14.10 STEU IV1 Register (STEUIV1R)

STEUIV1R STEU IV1 Register Offset 0xCD100

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	CC															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	CC															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	CB					CD	—									
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	R/W															
Reset	0x0000															

STEU IV1R is used during the initialization phase of both F8 confidentiality and F9 integrity algorithms. STEU IV1R must be written before processing of a new message begins. Once the initialization phase is completed, STEU IV1R is no longer used in normal processing of F8 or F9 modes.

Although this register is generally considered to be a context register, it does not need to be saved/restored during context switching in multi-session message processing. As with context registers, any byte of this register can be enabled or disabled during accesses.

Table 27-131. STEUIV1R Field Descriptions

Name	Reset	Description	Settings
CC 63–32	0	**Frame Independent Input Count** Stores the input count.	
CB 31–27	0	**Bearer Identity** Specifies the bearer identity.	
CD 26	0	**Direction** Specifies the message direction.	0 Inbound, decrypt 1 Outbound, encrypt
— 25–0	0	Reserved. Write to zero for future compatibility.	

Note: The user must ensure that fields of the STEUIV1 register are programmed correctly in accordance with the selected algorithm.

27.7.14.11 STEU ICV_In Register (STEUICVIR)

STEUICVIR — STEU ICV_In Register — Offset 0xCD108

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	ICV_In															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	ICV_In															
Type	R															
Reset	0x0000															

If ICV checking is required, the STEUICVIR holds the value to be compared with the computed F9 MAC value. The value must be written to STEUICVIR before the data size is written. The 32-bit written ICV occupies the lower half of the ICV_In register

27.7.14.12 STEU IV2 Register (STEUIV2R)

STEUIV2R — STEU IV2 Register — Offset 0xCD110

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	FRESH															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	FRESH															
Type	R/W															
Reset	0x0000															

The STEUIV2R holds the FRESH value that is used during the initialization phase of the F9 algorithm. This value is ignored when the F8 algorithm is selected. The FRESH value must be written before starting a new message to be processed with F9. Once the initialization phase is completed, STEUIV2R is not used during message processing. STEUIV2R does not need to be saved/restored during context switches.

27.7.14.13 STEU Context Register 1 (STEUCR1)

STEUCR1 — STEU Context Register 1 — Offset 0xCD118

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	MAC															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	MAC															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	MAC															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	MAC															
Type	R/W															
Reset	0x0000															

There are four 64-bit STEU context data registers that are used in F9 processing. These allow the core to interrupt F9 message processing and to resume at a later time. The Context Registers must be read when changing context and restored to their original values prior to resuming processing of an interrupted message. If the any of the STEU Context Registers is written during message processing, a Context Error is generated. All STEU Context Registers are cleared when a hard/soft reset or initialization is performed.

27.7.14.14 STEU Context Register 2 (STEUCR2)

STEUCR2 — STEU Context Register 2 — Offset 0xCD120

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	Z1															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	Z1															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Z2															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Z2															
Type	R/W															
Reset	0x0000															

There are four 64-bit STEU context data registers that are used in F9 processing. These allow the core to interrupt F9 message processing and to resume at a later time. The Context Registers must be read when changing context and restored to their original values prior to resuming processing of an interrupted message. If the any of the STEU Context Registers is written during message processing, a Context Error is generated. All STEU Context Registers are cleared when a hard/soft reset or initialization is performed.

27.7.14.15 STEU Context Register 3 (STEUCR3)

STEUCR3 — STEU Context Register 3 — Offset 0xCD128

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	Z3															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	Z3															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Z4															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Z4															
Type	R/W															
Reset	0x0000															

There are four 64-bit STEU context data registers that are used in F9 processing. These allow the core to interrupt F9 message processing and to resume at a later time. The Context Registers must be read when changing context and restored to their original values prior to resuming processing of an interrupted message. If the any of the STEU Context Registers is written during message processing, a Context Error is generated. All STEU Context Registers are cleared when a hard/soft reset or initialization is performed.

27.7.14.16 STEU Context Register 4 (STEUCR4)

STEUCR4 — STEU Context Register 4 — Offset 0xCD130

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	Z5															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	Z5															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	Total Message Bit Size															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Total Message Bit Size															
Type	R/W															
Reset	0x0000															

There are four 64-bit STEU context data registers that are used in F9 processing. These allow the core to interrupt F9 message processing and to resume at a later time. The Context Registers must be read when changing context and restored to their original values prior to resuming processing of an interrupted message. If the any of the STEU Context Registers is written during message processing, a Context Error is generated. All STEU Context Registers are cleared when a hard/soft reset or initialization is performed.

27.7.14.17 STEU LFSR State Registers 0–7 (STEULFSRSR[0–7])

STEULFSRSR0	STEU LFSR State Registers	Offset 0xCD138
STEULFSRSR1		Offset 0xCD140
STEULFSRSR2		Offset 0xCD148
STEULFSRSR3		Offset 0xCD150
STEULFSRSR4		Offset 0xCD158
STEULFSRSR5		Offset 0xCD160
STEULFSRSR6		Offset 0xCD168
STEULFSRSR7		Offset 0xCD170

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	S_{2i}															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	S_{2i}															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	$S_{(2i+1)}$															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	$S_{(2i+1)}$															
Type	R/W															
Reset	0x0000															

Note: i = the register index number.

There are eight 64-bit STEU LFSR registers that are used to record LFSR state during F8 processing. These registers must saved/restored when switching context in F8 mode.

27.7.14.18 STEU FSM State Registers 1 (STEUFSMSR1)

STEUFSMSR1 STEU FSM State Register 1 Offset 0xCD178

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	R0															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	R0															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	R1															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	R1															
Type	W															
Reset	0x0000															

The STEU FSM State Registers, are used to record the FSM state during F8 processing. These registers must be saved/restored when switching context in F8 mode.

27.7.14.19 STEU FSM State Register 2 (STEUFSMSR2)

STEUFSMSR2 STEU FSM State Register 2 Offset 0xCD180

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	R2															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	R2															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The STEU FSM State Registers, are used to record the FSM state during F8 processing. These registers must be saved/restored when switching context in F8 mode. Note that the lower half of FSM State Register 2 is not used.

27.7.14.20 STEU Key Data Registers 1–2 (STEUKDR[1–2])

STEUKDR1 STEU Key Data Registers 1–2 Offset 0xCD400
STEUKDR2 Offset 0xCD408

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	KRY															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	KRY															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	KRY															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	CK															
Type	W															
Reset	0x0000															

The STEU Key Data Registers together hold one 128-bit key that is used for initialization. STEUKDR1 (KEY-H) holds the upper 8 bytes of key data. STEUKDR2 (KEY-L) holds the lower 8 bytes. The STEU Key Data Registers must be written before message processing begins and cannot be written while the block is processing data, or a context error occurs. Reading from either of these registers always returns 0.

27.7.14.21 STEU Input FIFO/Output FIFO

STEU uses an input FIFO/output FIFO pair to hold data before and after processing. Normally, the channels control all access to these FIFOs. For core-controlled operation, a write to any area in the STEU FIFO address space enqueues data to the input FIFO, and a read from any area in the STEU FIFO address space dequeues data from the output FIFO.

Writes to the input FIFO go first to a staging register that supports 8-bit, 32-bit, or 64-bit accesses. When all 8 bytes of the staging register are written, the entire 8 bytes is automatically enqueued into the FIFO. If any byte is written twice between enqueues, an Address Error results. When writing the last portion of data, it is not necessary to write all 8 bytes. Any remaining input bytes in the staging register are automatically padded with zeros and forced into the input FIFO when the STEU End of Message Register is written.

The output FIFO supports 8-bit, 32-bit, or 64-bit read accesses. When all 8 bytes of the head 64-bit access are read, that 64-bit access is automatically dequeued from the FIFO so that the next 64 bits (if any) becomes available for reading. If any byte is read twice between dequeues, an Address Error results.

Overflows and underflows caused by reading or writing the STEU FIFOs are reflected in the STEU Interrupt Status Register. The STEU fetches data 64 bits at a time from the input FIFO. During F8 processing, the input data is XORed with the generated keystream and the results are placed in the output FIFO. The output size is the same as the input size. During F9 processing, the input data is hashed with the integrity key and the resulting MAC is placed in the Data Out Register.

The STEU input FIFO and output FIFO are located at offset 0xCD800–0xCDFFF.

27.7.15 RNGU Registers

27.7.15.1 RNGU Mode Register (RNGMR)

RNGMR — RNGU Mode Register — Offset 0xCA000

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The RNGU Mode Register is a writable location but all mode bits are currently reserved. It is documented for the sake of consistency with the other EUs.

27.7.15.2 RNGU Data Size Register (RNGDSR)

RNGDSR RNGU Data Size Register Offset 0xCA010

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The RNGU Data Size Register is used to tell the RNGU to begin generating random data. The actual content of the Data Size Register does not affect the operation of the RNGU. After a reset and prior to the first write of data size, the RNGU builds entropy without pushing data onto the FIFO. Once the Data Size Register is written, the RNGU begins pushing data onto the FIFO. Eight bytes (64 bits) of data is pushed onto the FIFO every 112 cycles until the FIFO is full. The RNGU then attempts to keep the FIFO full.

27.7.15.3 RNGU Reset Control Register (RNGRCR)

RNGRCR — RNGU Reset Control Register — Offset 0xCA018

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R/W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R/W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R/W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—													RI	MI	SR
Type	R/W															
Reset	0x0000															

The RNGRCR contains three reset options specific to the RNGU. 27-132 describes RNGRCR fields.

Table 27-132. RNGRCR Field Descriptions

Name	Reset	Description	Settings
— 63–3	0	Reserved. Write to zero for future compatibility.	
RI 2	0	**Reset Interrupt** Setting this bit causes RNGU interrupts signalling done and error to reset. It further resets the state of the RNGU Interrupt Status Register.	0 No reset. 1 Reset interrupt logic.
MI 1	0	**Module Initialization** Setting this reinitializes the RNGU to accept another request without forcing the internal control machines and the output FIFO to reset, which would invalidate stored random numbers or require reinvocation of a warm-up period. Module initialization is almost the same as a software reset except that the Interrupt Mask Register remains unchanged.	0 No reset 1 Reset most of RNGU
SR 0	0	**Software Reset** Setting this bit is functionally equivalent to a hardware reset (asserting the $\overline{\text{HRESET}}$ pin), but the reset is restricted to the RNGU. All registers and internal states are returned to the defined reset state.	0 No reset 1 Full RNGU reset.

27.7.15.4 RNGU Status Register (RNGSR)

RNGSR — RNGU Status Register — Offset 0xCA028

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—								OFL							
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—										HALT	—		EI	DI	RD
Type	R															
Reset	0x0000															

This RNGSR contains 5 fields that reflect the state of the RNGU internal signals. The RNGSR is read-only. Writing to this location results in an address error being reflected in the RNGISR. **Table 27-133** describes RNGSR fields.

Table 27-133. RNGSR Field Descriptions

Name	Reset	Description	Settings
— 63–24	0	Reserved. Write to zero for future compatibility.	
OFL 23–16	1	**Output FIFO Length** The number of 8-byte sets currently in the output FIFO.	
— 15–6	0	Reserved. Write to zero for future compatibility.	
HALT 5	0	**Halt** Indicates whether the RNGU is halted due to an error. **Note:** Because the error causing the RNGU to stop operating can be masked before reaching the Interrupt Status Register, the RNGU Interrupt Status Register is used to provide a second source of information regarding errors that prevent normal operation.	0 RNGU not halted 1 RNGU halted
— 4–3	0	Reserved. Write to zero for future compatibility.	
EI 2	0	**Error Interrupt** This status bit reflects the state of the error interrupt signal, as sampled by the controller Interrupt Status Register (**Section 27.7.4.6**, *Controller Interrupt Status Register (CISR),* on page 27-186).	0 RNGU is not signaling error 1 RNGU is signaling error

Table 27-133. RNGSR Field Descriptions (Continued)

Name	Reset	Description	Settings
DI 1	0	**Done Interrupt** This status bit, when high, reflects the state of the done interrupt signal as sampled by the Controller Interrupt Status Register (see **Section 27.7.4.6**, *Controller Interrupt Status Register (CISR)*, on page 27-186 for details).	0 No done interrupt. 1 RNGU signalling done.
RD 0	0	**Reset Done** This status bit, when high, indicates that the RNGU has completed its reset sequence. **Note:** The reset value of RD is 0, but it typically switches to 1 by the time a user checks the register, indicating the EU is ready for operation.	0 Reset in progress. 1 Reset done.

27.7.15.5 RNGU Interrupt Status Register (RNGISR)

RNGISR — RNGU Interrupt Status Register — Offset 0xCA030

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—			IE	—				ME	AE	—				OFU	—
Type	R															
Reset	0x0000															

The Interrupt Status Register indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the RNGU Interrupt Mask Register is zero (see **Section 27.6.9.6**, *RNGU Interrupt Mask Register*, on page 27-94). If the RNGU Interrupt Status Register is non-zero, the RNGU halts and the RNGU error interrupt signal is asserted to the controller (see **Section 27.6.9.5**, *RNGU Interrupt Status Register*, on page 27-93). In addition, if the RNGU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error bit is set in the Channel Status Register (see **Section 27.7.6.2**, *Channel Status Registers (CSR[1–4])*, on page 27-199) and generates a channel error interrupt to the controller. If the Interrupt Status Register is written from the core processor, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the

Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the RNGU Reset Control Register. **Table 27-134** describes RNGISR fields.

Table 27-134. RNGISR Field Descriptions

Name	Reset	Description	Settings
— 63–13	0	Reserved. Write to zero for future compatibility.	
IE 12	1	**Internal Error** An internal processing error was detected while generating random numbers.	0 No internal error detected. 1 Internal error.
— 11–8	0	Reserved. Write to zero for future compatibility.	
ME 7	0	**Mode Error** An illegal value was detected in the Mode Register.	0 Valid data. 1 Invalid data error.
AE 6	0	**Address Error** An illegal read or write address was detected within the RNGU address space.	0 No address error detected. 1 Address error detected.
— 5–2	0	Reserved. Write to zero for future compatibility.	
OFU 1	0	**Output FIFO Underflow** RNGU Output FIFO was read while empty.	0 No output FIFO underflow error detected. 1 Output FIFO underflow error detected.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.15.6 RNGU Interrupt Mask Register (RNGIMR)

RNGIMR — RNGU Interrupt Mask Register — Offset 0xCA038

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	R															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	R															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	R															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—			IE	—				ME	AE	—				OFU	—
Type	R															
Reset	0x1000															

The RNGU Interrupt Mask Register controls the result of detected errors. For a given error (as defined in **Section 27.6.9.5**, *RNGU Interrupt Status Register*, on page 27-93), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs, and

the Interrupt Status Register is not updated to reflect the error. If the corresponding bit is not set, then, upon detection of an error, the Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing. **Table 27-135** describes RNGU Interrupt Status Register fields.

Table 27-135. RNGIMR Field Descriptions

Name	Reset	Description	Settings
— 63–13	0	Reserved. Write to zero for future compatibility.	
IE 12	1	**Internal Error** An internal processing error was detected while generating random numbers.	0 Internal error enabled. 1 Internal error disabled.
— 11–8	0	Reserved. Write to zero for future compatibility.	
ME 7	0	**Mode Error** An illegal value was detected in the Mode Register.	0 Mode error enabled. 1 Mode error disabled.
AE 6	0	**Address Error** An illegal read or write address was detected within the RNGU address space.	0 Address error enabled. 1 Address error disabled.
— 5–2	0	Reserved. Write to zero for future compatibility.	
OFU 1	0	**Output FIFO Underflow** RNGU Output FIFO was read while empty.	0 Output FIFO underflow error enabled. 1 Output FIFO underflow error disabled.
— 0	0	Reserved. Write to zero for future compatibility.	

27.7.15.7 RNGU End_of_Message Register (RNGEOMR)

RNGEOMR — RNGU End_of_ Message Register — Offset 0xCA050

Bits	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
Field	—															
Type	W															
Reset	0x0000															

Bits	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Field	—															
Type	W															
Reset	0x0000															

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	—															
Type	W															
Reset	0x0000															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—															
Type	W															
Reset	0x0000															

The RNGU End_of_message is a writable location but serves no function in the RNGU. It is documented for the sake of consistency with the other EUs.

27.7.15.8 RNGU Entropy Registers 0–7 (RNGER[0–7])

The RNGU allows the user to input Entropy into the PRNG algorithm to modify the randomness of the RNGU. This group of registers are write only and all writes to these registers are ignored when the RNGU is busy. However, when the RNGU is IDLE (FIFO is full or RNGU has not yet been started), all data written to these registers is used to modify the internal XKEY structure. These registers cannot be written back to back, there must be a clock cycle in between writes, so that the RNGU can process all 64-bits of data, because the RNGU processes only 32-bits per cycle. The eight 8-byte Entropy Registers 0–7 are located from offset 0xCA400 to 0xCA43F.

27.7.15.9 RNGU Output FIFO

RNGU uses an output FIFO to collect periodically sampled random 64-bit-words, with the intent that random data always be available for reading. Normally, the channels control all access to this FIFO.

Note: For core processor-controlled operation, a read from anywhere in the RNGU FIFO address space dequeues data from the RNGU output FIFO. Such reads should be performed on an 8-byte basis regardless of the number of random number bits required. Partial reads by the core processor can leave the RNGU FIFO in a state that results in a channel error.

The output FIFO is readable by byte, 4-byte, or 8-byte accesses. When the first 8 bytes are read, that 8 bytes is automatically dequeued from the FIFO so that the next 8 bytes (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Underflows caused by reading or writing the RNGU output FIFO are reflected in the RNGU Interrupt Status Register. Also, a write to the RNGU output FIFO space is reflected as an addressing error in the RNGU Interrupt Status Register.

Note: The RNGU output FIFO is located at offset 0xCA800–0xCAFFF.